Automatic verification of counter systems with ranking function

Emmanuelle Encrenaz & Alain Finkel

LSV, ENS-Cachan & CNRS
Cachan, France
{*encrenaz,finkel*}*@lsv.ens-cachan.fr*

Abstract

The verification of final termination for counter systems is undecidable. For non flattable counter systems, the verification of this type of property is generally based on the exhibition of a ranking function. Proving the existence of a ranking function for general counter systems is also undecidable. We provide a framework in which the verification whether a given function is a ranking function is decidable. This framework is applicable to convex counter systems which admit a Presburger or a LPDS ranking function. This extends the results of [6]. From this framework, we derive a model-checking algorithm to verify whether a final termination property is satisfied or not. This approach has been successfully applied to the verification of a parametric version of the ZCSP protocol.

Keywords: Final Termination Property, Ranking function, Convex Counter Systems, Automatic Verification, Parametric Protocol ZCSP.

1 Introduction

While verifying a parametric protocol (ZCSP) with FAST [2], we came across an interesting problem. We had to verify a *final termination* property, expressing that the system will end in a given set of states in an unavoidable manner. Unfortunately, the class of counter systems modelling the protocol did not fit with the hypothesis under which FAST may automatically solve it: our model for ZSCP is neither flat nor trace-flattable.

Indeed, the final termination property is undecidable in the general case, and one has to consider some strong hypotheses to automate its verification. This termination problem is classically solved by exhibiting a ranking function; it has been actively studied in the last three years in the context of code analysis for imperative programs containing loops with integer variables. In this context, [15] presents a complete method for the synthesis of linear ranking functions on the restricted class of single path loops. This result has been recently extended in [7] to (single path) nested loops and is implemented in the tool TERMINATOR [8], devoted to the analysis of C code for hardware device drivers. A complementary approach is presented in [14]. A semi-algorithm based on *region graphs* is proposed; it applies to exclusive multiple-path loops and is implemented in the PONES tool, devoted to the verification of Java programs. [6] synthesizes linear ranking functions for a larger class of systems: Integer-variable loops with multiple paths with non-exclusive guards. The synthesis is based on an enumeration of all linear functions (represented as Presburger formulas). The method is complete (if such a linear function exists, the procedure will eventually exhibit it), however, this work does not consider parameters.

Our contributions. We revisit the ranking function synthesis problem in the context of (possibly non deterministic) counter systems . We distinguish between the problem of the Existence of a ranking function and the problem of Verification whether a function in a given class is a ranking function. We first recall that the existence of a recursive ranking function is undecidable, but it becomes decidable when considering trace-flattable counter systems. Similarly, verifying whether a recursive function is a ranking function is undecidable although verifying a Presburger definable ranking function is decidable. Unfortunately, ZCSP does not admit any Presburger definable ranking function, but it admits a ranking function definable in a Presburger extension allowing multiplication with a unique parameter.

M. Bozga, R. Iosif and Y. Lakhnech showed in [5] that Linear Parametric Diophantine Systems (LPDS) are effectively solvable. LPDS strictly extend the existential fragment of Presburger arithmetic in allowing the multiplication of a variable with a unique parameter p. We prove that verifying if a counter system (using $Presb^{\exists}$-definable linear functions and having a $Presb^{\exists}$-definable reachability set) satifies a LPDS definable ranking function is decidable.

From this result, we derive a procedure to automatically synthesize either Presburger-definable ranking functions or LPDS-definable ranking functions. The procedure will enumerate potential ranking functions and check them. The procedure terminates if and only if a Presburger-definable or a LPDS-definable ranking function exists. The proposed approach is used to verify a final termination property of the protocol ZCSP. The method extends the aforementionned works since our hypothesis are as general as [6] (which are larger than [14] and [7]), and the class of ranking functions we synthesize is larger than [6].

The exhibited ranking function could not have been found with the cited methods or tools, since it required the most relaxed hypothesis (multiple-path loop with non-exclusive guards), and did not admit any Presburger linear ranking function. In particular, when analyzing multi-threaded programs, TERMINATOR focuses on the "thread termination" property, which is not the property we want to verify.

Organisation of the paper. A preliminary section collects some useful notions about flat and flattable counter systems. Sections 3 and 4 present an abstraction of the ZCSP protocol as a counter system and the verification of safety properties that have been achieved with FAST. Section 5 defines a method to prove the final termination of a counter system with the automatic synthesis of a ranking function. In Section 6, this method is illustrated on the model of ZCSP. The appendix gives details about the ZCSP protocol and presents the derived counter system. The description of the ZCSP protocol and complete proofs of propositions in Sections 4, 5 and 6 are given in the appendix of the long version of the paper on the web pages of the authors.

2 Preliminaries

2.1 Counter systems

We recall that *Presburger arithmetic* is the first order theory of the structure $\langle \mathbb{N}, +, = \rangle$. Given a Presburger formula ϕ with free variables belonging to the set C of counters and $\mathbf{a} \in \mathbb{N}^C$, we write $\mathbf{a} \models \phi$ if ϕ is true for the valuation $\mathbf{a}$. A set $X \subseteq \mathbb{N}^n$ is said to be *Presburger definable* iff there is a Presburger formula $\psi(\mathbf{x})$ with free variables $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ such that $X = \{\mathbf{a} \in \mathbb{N}^n : \mathbf{a} \models \psi(\mathbf{x})\}$. This can be extended without problems to $\mathbb{Z}^n$. Presburger arithmetic is known to be decidable and therefore, all the problems in the forthcoming sections that can be reduced to Presburger arithmetic are decidable. We recall that the set of polyhedral convex sets is exactly equal to the set of $Presb^{\exists}$ definable sets, where $Presb^{\exists}$ is the Presburger existential fragment (without modulo). Abusing notation: we will denote by ϕ the set defined by the formula ϕ. A *Presburger function* is a partial function definable by a Presburger formula. A *Presburger-linear function* f is a Presburger function which can be represented by a tuple $(A, \mathbf{b}, \phi)$ where A is a square matrix in $\mathbb{N}^{C \times C}$, $\mathbf{b} \in \mathbb{Z}^C$ and ϕ is a Presburger formula such that $f(\mathbf{a}) = A.\mathbf{a} + \mathbf{b}$ for every $\mathbf{a} \models \phi$ (ϕ is a formula representing the domain of f, also denoted by $dom(f)$). We denote by Σ_C the set of such functions.

Definition 2.1 A *counter system* is a graph whose edges are labeled with Presburger linear functions, that is a tuple $CS = \langle Q, E \rangle$ where $E \subseteq Q \times \Sigma_C \times Q$.

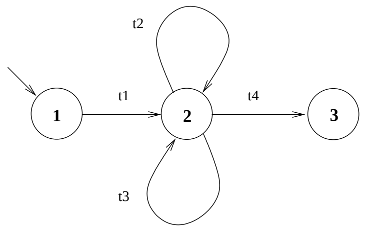
With a counter system $CS = \langle Q, E \rangle$, we associate the transition system $TS(CS) = \langle Q \times \mathbb{N}^C, \rightarrow \rangle$ defined by $(q, \mathbf{a}) \rightarrow (q', \mathbf{a}')$ if there is a transition $q \xrightarrow{f} q'$ in E such that $\boldsymbol{a}' = f(\boldsymbol{a})$. A *simple cycle* in a graph $G = \langle Q, E \rangle$ is a closed path (where the initial and final vertices coincide) with no repeated edge. G is said to be *flat* if every $q \in Q$ belongs to at most one simple cycle. A counter system CS is said to have *the finite monoid property* if the multiplicative monoid generated by the matrices used in its labels is finite. Note that for a counter system $CS = \langle Q, E \rangle$, the control states can be encoded as positive integers (ie $Q \subseteq \mathbb{N}$) and then the set of configurations is represented by $\mathbb{N}^{|C|+1}$.

Theorem 2.2 *[11] Let CS be a flat counter system $\langle Q, E \rangle$ with the finite monoid property and $TS(CS) = \langle \mathbb{N}^{|C|+1}, \rightarrow \rangle$ its associated transition system. Then the reflexive and transitive closure $\rightarrow^*$ of the reachability relation is effectively Presburger definable.*

In the following, we will assume that the set of states is in $\mathbb{N}^n$. In [9], a temporal logic for counter systems –FOPCTL*(Pr)– is introduced. The model-checking of a flat counter system with the finite monoid property and a formula in FOPCTL*(Pr)is decidable.

2.2 Model-checking for flattable systems

Flat counter systems have numerous desirable properties, however, realistic systems are rarely flat. It is interesting to consider larger classes of systems – called flattable counter systems – that are reducible to flat counter systems via graph homomorphism [9].



$$t_1 = [TRUE] \rightarrow \{x' := p1, y' := p2\}$$
$$t_2 = [x > 0] \rightarrow \{x' := x - 1, y' := y\}$$
$$t_3 = [x = 0 \wedge y = 0] \rightarrow \{x' := p1, y' := y - 1\}$$
$$t_4 = [x = 0 \wedge y = 0] \rightarrow \{x' := x, y' := y\}$$

Fig. 1. A post*-flattable but not trace-flattable system

Definition 2.3 Let $CS = \langle Q, E\rangle$ and $CS' = \langle Q', E'\rangle$ be two counter systems, having the finite monoid property, of the same dimension, h be a function $h : Q' \rightarrow Q$, $q \in Q$ and $q' \in Q'$. $\langle CS', q'\rangle$ is a h-**flattening** of $\langle CS, q\rangle$ iff $h(q') = q$, CS' is flat, and whenever $\langle s, f, s'\rangle \in E'$, we have $\langle h(s), f, h(s')\rangle \in E$.

When $\langle CS', q'\rangle$ is a h-**flattening** of $\langle CS, q\rangle$, CS can be viewed as an abstraction of CS'. The tool FAST [2] generates flattenings via an exhaustive search algorithm. Several flattenings are defined and for each of them, the preserved sub-classes of FOPCTL*(Pr)formulas are established. The most common relationship between CS and CS' is the equality of reachability sets (leading to the notion of post*-flattening).

Let $CS = \langle Q, E\rangle$ be a counter system. The reachability sets from a configuration and from a set of Presburger definable configurations are defined as follows:

- $\text{post}^*_{TS(CS)}(\langle q, \mathbf{a}\rangle) \overset{\text{def}}{=} \{\langle q', \mathbf{a}'\rangle \in Q \times \mathbb{N}^C : \langle q, \mathbf{a}\rangle \rightarrow^* \langle q', \mathbf{a}'\rangle\}$.
- $\text{post}^*_{TS(CS)}(q, \psi(\mathbf{x})) \overset{\text{def}}{=} \bigcup_{\mathbf{a} \models \psi(\mathbf{x})} \text{post}^*_{TS(CS)}(\langle q, \mathbf{a}\rangle)$.

Definition 2.4 (from [9]) $\langle CS, q'\rangle$ is a h-**post*-flattening** (post*-flattening for short) of $\langle CS, q\rangle$ with respect to ψ) iff $\text{post}^*_{TS(CS)}(q, \psi) = h(\text{post}^*_{TS(CS')}(q', \psi))$ and CS' is a h-flattening of CS (h is naturally extended to states of $TS(CS)$); we say that $\langle CS, q\rangle$ is post*-flattable.

Post*-flattening preserves reachability properties [9]. Intuitively, a system CS' is a trace-flattening (cf. Appendix A) of a system CS if CS' is a h-flattening of CS and if the set of traces of CS is equal to the image by h of the set of traces of CS'. Trace-flattening preserves the LTL fragment of FOPCTL*(Pr)which is decidable for trace-flattable counter systems. As a consequence, the final termination problem can be expressed as a LTL formula hence it is decidable for trace-flattable counter systems.

Example 2.5 The system CS_1 described in Fig. 1 is not flat, but it is *post**-flattable from the initial configuration $Init_0 = \mathbb{N}^4$. The reachability set from $Init_0$ is obtained by the flat trace $t_1.t_2^*.t_3^*.t_4$ which can be computed by acceleration [11].

Moreover, CS_1 is *not* trace-flattable [9]. The system produces a non-finite union of flat traces (the size of the union depends on parameter p_2, which is unbounded).

In practice, the *post**flattable framework works quite well for verifying safety properties (see e.g. [11],[3],[2]). But, realistic systems are rarely trace-flattable. Hence, the proof of final termination must, in general, rely on another approach.

3 A counting abstraction of protocol ZCSP

3.1 Presentation of the protocol ZCSP

Protocol ZCSP (for Zero-Copy Secure Protocol) is a communication protocol implemented in the MPC parallel computer [10]. In essence, ZCSP protocol is a variant of BRP protocol that has been extensively studied (for instance, see [1]). In ZCSP several messages may be emitted before the respective acknowledgments are received; the acknowledgements may be received out of order; emitted messages have to be stored up to the reception of their own acknowledgment and those of their predecessors. This storage induces a greater complexity than BRP.

3.2 Desirable properties

P1. The number of table entry is constant (and equal to $TMAX$).
P2. At a given time, there is never more than one message under re-emission.
P3. If there is no re-emission, the counter of re-emission is set to 0.
P4. Each lost message will be re-emitted.
P5. Some message re-emission will reach the maximal retransmission bound.
P6. No re-emitted message oversteps the maximal retransmission bound.
P7. If the table contains any number of message to be emitted, and no new message is eventually inserted, then the table and channels will unavoidably become empty.

3.3 A counting abstraction of ZCSP

We present a counter system abstraction of ZCSP. The system has been abstracted in two directions : messages are atomic, and their identity is not not represented.

The counter system contain 14 counters. With this abstraction, messages in the table are not identified by their entry-index, but rather by their state. The content of the storage table is modeled as a set of five counters c_1, c_2, c_3, c_4, c_5 indicating the number of messages in each corresponding category. The channel StoR, transmitting messages from the sender to the receiver, is modelled as two counters c_6 and c_7, distinguishing the first emission of a message from a re-emission. In the same way, the channel RtoS, transmitting acknowledgments from the receiver to the emitter is modelled by two counters c_8 and c_9. The timeout occurrences are modelled as two counters c_{10} and c_{11}. The current number of re-emission is modeled as a counter c_{12}. Counters c_{13} and c_{14} contain resp. the maximal retransmission number and the number of entry in the storage table.

We denote Z the counter system which is composed of a unique local state and 16 self-looping transitions. Every state s of Z is a tuple $(s_1, s_2, ...s_{14}) \in \mathbb{N}^{14}$;

Proposition 3.1 *Forall* $1 \leq i \leq 7$, *if* $< Z \models P_i >$ *then* $< ZCSP \models P_i >$

Proof. (sketch) The abstraction represents an overapproximation of the set of behaviours of ZCSP: files are represented as counters and bounds on files are relaxed. Moreover, messages are now atomic. This coarser representation does not miss any interleavings since in ZCSP, packets of a given message are sent atomically. □

4 Verification of safety properties with FAST

The counter system Z is not flat. $Init$ is the initial state, defined as follows: $Init = \{s \in \mathbb{N}^{14} \mid s_1 + s_2 + s_3 + s_4 + s_6 + s_7 + s_8 + s_9 + s_{10} + s_{11} + s_{12} = 0 \ \wedge \ s_5 = s_{13} \ \wedge \ s_{13} > 0 \ \wedge \ s_{14} > 0\}$. $Init$ represents the set of configurations when the pending message table is empty (all entries are free), the channels $StoR$ and $RtoS$ are empty, there is no pending timeout, and the re-emission counter is set to 0.

Proposition 4.1 *$(Z, Init)$ is post*-flattable.*

Hence reachability properties can be automatically checked. Properties **P1** to **P3**, **P4'** which is a relaxation of **P4**, and **P5** to **P6** were automatically verified with FAST.

We now concentrate on the property **P7**: "If the table contains any number of message to be emitted, and no new message is eventually inserted, then the table and channels will unavoidably become empty". This property expresses a final termination, it is not reducible to a reachability property but is expressible in LTL or CTL. Unfortunately, the language of $(Z, Init)$ contains a sequence of the form $(ab^pc)^n$, hence:

Proposition 4.2 *$(Z, Init)$ is* not *trace-flattable.*

Hence final termination properties cannot be checked by an automatic trace-flattening of $(Z, Init)$. The automatic synthesis of a ranking function is an alternative.

5 Proving final termination with automatic synthesis of ranking functions

5.1 Ranking function for termination

Let us note CS_{presb} the class of counter systems CS such that the relation $\rightarrow^*_{TS(CS)}$ is effectively Presburger definable. We denote CS_{post^*} (resp. CS_{trace}) the set of counter systems CS, with an initial Presburger set $Init$, such that it is post*-flattable (resp. trace-flattable). Let us remark that: $CS_{trace} \subseteq CS_{post^*} \subseteq CS_{presb}$.

Definition 5.1 Let TS be a transition system and $Init$ and $Final$ two sets. $< TS, Init, Final >$ is deadlock-free if $\forall s \in post^*_{TS}(Init) \setminus Final$, $post_{TS}(s) \neq \emptyset$.

Proposition 5.2 *Given a counter system $CS \in C_{presb}$ and two Presburger sets $Init$ and $Final$, the deadlockfree property of $< TS(CS), Init, Final >$ is decidable.*

When CS is flat with a finite monoid, then the set $post^*_{TS(CS)}(Init)$ is an effective Presburger-definable set, hence:

Corollary 5.3 *Given a flat CS with a finite monoid and two Presburger sets Init and $Final$, the deadlockfree property of $< TS(CS), Init, Final >$ is decidable.*

Ranking functions are often used for proving termination. A general ranking function f is a function from the set S of states into an ordered set $(N, \prec)$ such that there do not exist infinite strictly decreasing sequence in N. For counter systems CS of dimension n, we will study recursive functions from $\mathbb{N}^{n+1}$ into $\mathbb{N}$. This is not a restriction because to every ranking function $f : \mathbb{N}^{n+1} \rightarrow \mathbb{N}^k, k \geq 1$ one may associate another ranking function $f' : \mathbb{N}^{n+1} \to \mathbb{N}$ such that $f'(x) = y_1 + y_2 + ... + y_k$ with $f(x) = (y_1, y_2, ..., y_k)$.

Definition 5.4 Let us consider a transition system $TS(CS) =< S, \to >$ with two sets of configurations $Init, Final \subseteq S$ and a function $f : S \to \mathbb{N}$. We say that a recursive function f is a ranking function of $(TS(CS), Init, Final)$ if $\forall x, x' \in post^*_{TS(CS)}(Init) \setminus Final$, $x \to x'$ implies $f(x') < f(x)$.

Proposition 5.5 *For any transition system* $TS(CS) =< S, \to >$ *equipped with two sets* $Init, Final$*, such that* $(TS(CS), Init, Final)$ *is deadlockfree, we have* $< TS(CS), Init > \models AF\ \ Final$ *iff there exists a ranking function for* $< TS(CS), Init, Final >$*.*

5.2 Decidability of the ranking function property

Given a class C of transition systems (with S as set of states), a class X of recursive sets and a class F of recursive functions from S to $\mathbb{N}$, we distinguish two problems associated with each triple (C, X, F):

(i) The Existence Ranking Problem ERP(C,X,F).
Input: Given a transition system $TS =< S, \to >$ in C, two sets of configurations $Init, Final \in X$.
Output: To decide whether there exists a ranking function $f \in F$ for $< TS, Init, Final >$?

(ii) The Verification Ranking Problem VRP(C,X,F).
Input: Given a transition system $TS =< S, \to >$ in C, two sets of configurations $Init, Final \in X$ and a function $f \in F$.
Output: Is f a ranking function of $< TS, Init, Final >$?

We denote X_{presb} (resp. X_{conv}) the set of Presburger-definable sets (resp. the set of Presburger polyhedral convex sets) and F_{rec} (resp. F_{presb} and $F_{presblin}$) the set of recursive functions (resp. Presburger functions and Presburger-linear functions).

From the fact that liveness properties are undecidable for post*-flattable CS with finite monoid, we deduce that:

Proposition 5.6 *The Existence Ranking Problem ERP(*$CS_{post^*}, X_{presb}, F_{rec}$*) is undecidable.*

From the fact that the LTL model-checking of trace-flattable CS with a finite monoid is decidable [9], we may deduce:

Proposition 5.7 *The Existence Ranking Problem ERP(*$CS_{trace}, X_{presb}, F_{rec}$*) is decidable.*

There exists a reduction of the problem of testing whether a recursive function is decreasing to the VRP$(CS_{post^*}, X_{presb}, F_{rec})$. We build a CS of dimension n, with $Init = 0$ as the initial state; CS has an unique local state and for every counter c_i, there exists a transition $t_i : c_i := c_i + 1$. This counter system is not flat but it is post*-flattable and its reachability set is equal to $post^*_{TS(CS)} = \mathbb{N}^C$. Now the condition for a recursive function f from $\mathbb{N}^C$ into $\mathbb{N}$ to be a ranking function of $< TS(CS), Init = 0, Final = \mathbb{N}^C >$, remains to say that f is strictly decreasing. And this last problem is undecidable [12]. Hence we obtain:

Proposition 5.8 *The Verification Ranking Problem VRP($CS_{post^*}, X_{presb}, F_{rec}$) is undecidable.*

Although this last problem was undecidable, there exists a decidable sufficient condition for any counter system CS and any Presburger function f; as a matter of fact, one may always decide the satisfiability of the following Presburger formula: $(\forall x, x' \in \mathbb{N}^{C+1}, x \rightarrow x'$ implies $f(x') < f(x))$. f is called a *absolute ranking* function.

The VRP becomes decidable if one restricts the class of functions to be Presburger-definable. The condition to be a ranking function can be coded into a Presburger formula ϕ and we obtain that the VRP is true iff ϕ is satisfiable then it becomes decidable.

Proposition 5.9 *The Verification Ranking Problem VRP($CS_{presb}, X_{presb}, F_{presb}$) is decidable.*

This last result may suggest to enumerate (fairly and efficiently) all Presburger functions and to test whether every Presburger function is a ranking function. This strategy will find a ranking function if there exists one Presburger ranking function. In the other case, the computation will not terminate. In particular, it may exist a non-Presburger ranking function.

For instance, let us suppose that there only exists a ranking function which uses some kind of multiplication between variables, typically between a parameter (i.e. a variable which is never modified) and a variable. In the general case, the VRP would be undecidable for this sort of functions. Let us first recall that Linear Diophantine Systems can be written as a boolean combination of linear equations of the form: $\Sigma_{i=1}^{n} e_i.x_i + e_0 = 0$ where all $e_i \in \mathbb{Z}$. Their set of solutions are a Presburger set, and more precisely, a polyhedral convex Presburger set. It is possible to extend this sort of systems to Linear Parametric Diophantine Systems in allowing some multiplications between one variable and the unique parameter.

Let us denote $\mathbb{Z}_k[p]$ the set of polynoms of maximum degree k, whose *unique* variable is p. A Linear Parametric Diophantine System (LPDS) [5] is a Linear Diophantine System that can be written as a boolean combination of equations of the form: $\Sigma_{i=1}^{n} e_i.x_i + e_0 = 0$ where all $e_i \in \mathbb{Z}_k[p]$. From [5] (Theorem 2) one knows that the satisfiability problem for LPDS is decidable.

Let us note that LPDS strictly extends the existential fragment of Presburger arithmetic. On the other hand, no universally quantified Presburger formula is allowed in LPDS. A formula which is both in Presburger and in LDS is in $Presb^{\exists}$. We now define LPDS functions allowing a kind of *multiplication* between any variable *and* the unique parameter p.

Definition 5.10 A LPDS function is a function definable by a LPDS.

We denote by F_{LPDS} the set of LPDS functions. For example, $f(\mathbf{x}) = c_i.\mathbf{x} + d_j$ with c_i in $\mathbb{Z}_k[\mathbf{p}]^{\mathbf{C}}$ and d_j in $\mathbb{Z}_k[\mathbf{p}]$ is a LPDS function. Let us remark that every (integer) linear function with a polyhedral convex domain is a LPDS function without parameter. The converse is obviously false.

Definition 5.11 A counter system $(CS, Init)$ is said *convex* if the domain of each Presburger-linear function of CS is polyhedral convex and if $post^*_{TS(CS)}(Init)$ is polyhedral convex.

Let us denote by CS_{conv} the set of convex counter systems. From the fact that given a Presburger formula, one may decide if it is equivalent to a formula in $Presb^{\exists}$, we deduce :

Proposition 5.12 *The convex property is decidable for counter systems with a effective Presburger reachability set.*

Proposition 5.13 *The VRP($CS_{conv}, X_{conv}, F_{LPDS}$) is decidable.*

5.3 Model-checking procedure for counter systems

The model-checking procedure consists in enumerating functions, and for each fonction, check if it satisfies the ranking function condition.

First we have to find the parameters. We first try to find among the variables those which are in fact parameters, (i.e. parameters are variables that are never modified by all the functions of the counter system). If there are no parameters, enumerate Presburger functions and test. Else, one computes the set of parameters. We may test if a variable x is a parameter by the help of the Presburger formula associated with each transition of the counter system (or in computing post*) and in verifying that the variable x never changes its value. Then for every parameter , enumerate the LPDS functions and test whether it is a ranking function.

```
Procedure Model-Check($CS$:counter system; $Init, Final$:  two polyhedral
convex sets)
 1.  Compute with FAST $Post^*_{TS(CS)}(Init)$;
 2.  Compute $Deadlock = \bigcap_{t \in E} \neg dom(t)$;
 3.  if $Post^*_{TS(CS)}(Init) \cap Deadlock \neq \emptyset$ return FALSE;
 4.  Compute the set $P$ of parameters of $CS$;
 5.  If $P = \emptyset$ then
   (a) Enumerate all Presburger functions $f$
   (b) If $f$ is a ranking function for $< TS(CS), Init, Final >$ then return
       TRUE else goto 5(a)
 6.  Else for every parameter $p \in P$ enumerate all LPDS functions $f$
   (a) If $CS$ is a convex counter system then
     a.1.  If $f$ is a ranking function for $< TS(CS), Init, Final >$ then
           return TRUE
     a.2.  else goto 6.
   (b) Else If $f$ is a absolute ranking function then return TRUE else
       goto 6.
```

Proposition 5.14 *If procedure* `Model-Check` *terminates then* $CS, Init \models AF\ \ Final$.

The converse is not true : a system may have a ranking function not being in F_{LPDS} neither in F_{presb}. In this last case, the procedure `Model-check` will not find it and will not terminate.

6 Proving final termination in finding a ranking function

Let us return back to the verification of property **P7** of system Z. In Z, the emission of new messages is modeled by transition t_1. We consider Z' being the system Z *without transition t1*. We denote $Init'$ the set of states representing the non empty table.

$Init' = post^*_{t_1}(Init) = \{s_1 \leq s_{13} \;\wedge\; s_2+s_3+s_4+s_5+s_7+s_8+s_9+s_{10}+s_{11}+s_{12} = 0 \;\wedge\; s_6 = s_1 \;\wedge\; s_{13} > 0 \;\wedge\; s_{14} > 0\}$.

We denote $Final'$ the unavoidable set of states in Z' from $Init'$. $Final'$ represents a table with all entries being free and channels being empty. It corresponds to the set of states $Init$.

Property **P7** may be now expressed as : $\forall s \in Post^*_{t_1}(Init), s \;\models\; AF \;\; Final'$. This can be rephrased in Z' : $< Z', Init' \;\models\; AF \;\; Final' >$.

To prove this property, we apply the algorithm defined in Sec. 5.3.

Remark : We can see that $Init'$, $Final'$ and the domain of each each transition of Z' are convex. Even if we can theoretically decide whether Z' is convex or not, we were not able to automatically test it; it will be done once the implementation of the result of [13] will be achieved.

Here are the successive steps of the `Model-Check(`Z'`,`$Init'$`,`$Final'$`)`:

step 1. Compute $Post^*_{Z'}(Init')$

step 2. Compute $Deadlock = \cap_{2\leq i\leq 16} \neg dom(t_i)$

step 3. $post^*_{Z'}(Init') \setminus Final' \bigcap Deadlock = \emptyset$

step 4. $P = \{c_{13}, c_{14}\}$.

step 5. As $P \neq \emptyset$, we directly jump to step 6.

step 6. Consider parameter c_{14} in P and enumerate the LPDS function f with respect to parameter c_{14}.

step 6.a. We don't know whether Z' is convex.

step 6.b. For each f, decide whether f is a absolute ranking function.

Let f be the following LPDS function from $\mathbb{N}^{14}$ to $\mathbb{N}$:

$f(\mathbf{s}) = (3.s_{14}+5)(3.s_6+2.s_8+s_{10}) \;+\; (3.s_{14}+4).s_4 \;+\; (3.s_7+2.s_9+s_{11}+3.s_{12}) \;+\; 2.s_2 \;+\; (s_{13} - s_5)$

Proposition 6.1 *f is a LPDS absolute ranking function for $< Z', Init', Final' >$.*

We also prove that:

Proposition 6.2 *$< Z', Init', Final' >$ does not admit a linear ranking function.*

7 Conclusion and perspectives

We characterize the classes of systems for which the proposed analysis is feasable. We propose a model-checking algorithm to analyse the final termination property of counter systems. Our procedure is complete: the procedure terminates iff a ranking function of a given class exists. Our results extend the class of Bradley's ranking functions and are similar to those obtained with TERMINATOR. It is not clear whether the last version of TERMINATOR (announced in June 2007 and not available) analyzing multi-threaded programs would terminate on ZCSP.

In order to automate the model-checking procedure, several points have to be solved.

- to have an efficient procedure for solving LPDS. To the best of our knowledge, no such dedicated tool exists.

- to have an efficient enumeration scheme of potential ranking functions (either Presburger or LPDS definable). One could follow Bradley's approach to prune the enumeration space.

- to determine whether $Post^*(Init)$ is a polyhedral convex set. A way to proceed consists in translating the symbolic representation of $Post^*(Init)$ into a Presburger formula, and then to check whether this formula is convex or not.

References

[1] P. Abdulla, A. Annichini, and A. Bouajjani. Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In *Tools and Algorithms for the Consctruction and Analysis of Systems (TACAS)*, volume 1579 of *Lecture Notes in Computer Science*, pages 208–223. Springer, 1999.

[2] S. Bardin, A. Finkel, J. Leroux, and L. Petrucci. Fast: Fast acceleration of symbolic transition systems. In *CAV 2003,* 15^{th} *International Conference on Computer-Aided Verification, Boulder, CO, USA*, volume 2725 of *Lecture Notes in Computer Science*, page short paper. Springer, 2003.

[3] S. Bardin and L. Petrucci. From pnml to counter systems for accelerating petri nets with fast. In *Workshop on Interchange Format for Petri Nets, Bologna, Italy*, pages 26–40, 2004.

[4] V. Beaudenon, E. Encrenaz, and J.-L. Desbarbieux. Design validation of zcsp with spin. In *IEEE* 3^{rd} *Int. Conf. on Application of Concurrency to System Design (ACSD)*, pages 102–110. IEEE Computer Society Press, 2003.

[5] M. Bozga, R. Iosif, and Y. Lakhnech. Flat parametric counter automata. In *Automata, Languages and Programming,* 33^{rd} *International Colloquium, ICALP 2006, Venice, Italy, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 577–588. Springer, 2006.

[6] A. Bradley, Z. Manna, and B. Sipma. Termination analysis of integer linear loops. In *CONCUR 2005 - Concurrency Theory, 16th International Conference, CONCUR 2005, San Francisco, CA, USA,*, volume 3653 of *Lecture Notes in Computer Science*, pages 488–502. Springer, 2005.

[7] B. Cook, A. Podelski, and A. Rybalchenko. Termination proofs for systems code. In *PLDI, Proceedings of the ACM SIGPLAN 2006 Conference on Programming Language Design and Implementation, Ottawa, Ontario, Canada, June 11-14, 2006*, pages 415–426, 2006.

[8] B. Cook, A. Podelski, and A. Rybalchenko. Terminator: Beyond safety. In *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA*, pages 415–418, 2006.

[9] S. Demri, A. Finkel, V. Goranko, and G. van Drimmelen. Towards a model-checker for counter systems. In *ATVA 2006,* 4^{th} *International Symposium on Automated Technology for Verification and Analysis, Beijing, Rep. of China*, volume 4218 of *Lecture Notes in Computer Science*, pages 493–507. Springer, 2006.

[10] J.-L. Desbarbieux, O. Glck, A Zerrouki, A. Fenyo, A Greiner, F. Wajsbrt, C. Spasevski, F. Silva, and E. Dreyfus. Protocol and performance analysis of the mpc parallel computer. In 15^{th} *Int. Parallel and Distributed Processing Symposium*, page 52. ACM, 2001.

[11] A. Finkel and J. Leroux. How to compose presburger accelerations: Application to broadcast protocols. In 22^{th} *Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS), Kanpur, India*, volume 2556 of *Lecture Notes in Computer Sciences*, pages 145–156. Springer, 2002.

[12] Alain Finkel, Pierre McKenzie, and Claudine Picaronny. A well-structured framework for analysing Petri net extensions. *Information and Computation*, 195(1-2):1–29, November 2004.

[13] J. Leroux. A polynomial time presburger criterion and synthesis for number decision diagrams. In 20^{th} *IEEE Symp. Logics in Computer Science (LICS)*, pages 147–156. IEEE Computer Society Press, 2005.

[14] S. Leue and W. Wei. A region graph based approach to termination proofs. In *TACAS, Tools and Algorithms for the Construction and Analysis of Systems, 12th International Conference, TACAS 2006 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25 - April 2, 2006*, pages 318–333, 2006.

[15] A. Podelski, A.and Rybalchenko. A complete method for the synthesis of linear ranking functions. In *Verification, Model Checking, and Abstract Interpretation,* 5^{th} *International Conference, VMCAI 2004,*, volume 2937 of *Lecture Notes in Computer Science*, 2004.