Liveness and boundedness of synchronous data flow graphs
Ghamarian, A.H.; Geilen, M.C.W.; Basten, A.A.; Theelen, B.D.; Mousavi, M.; Stuijk, S.

Published: 01/01/2006

Document Version
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the author's version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain.
• You may freely distribute the URL identifying the publication in the public portal?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 11. Dec. 2018
Liveness and Boundedness of Synchronous Data Flow Graphs

AmirHossein Ghamarian, Marc Geilen, Twan Basten, Bart Theelen, MohammadReza Mousavi, Sander Stuijk
Liveness and Boundedness of Synchronous Data Flow Graphs

Eindhoven University of Technology, Electronic Systems Group
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
a.h.ghamarian@tue.nl

Abstract

Synchronous Data Flow Graphs (SDFGs) have proven to be suitable for specifying and analyzing streaming applications that run on single- or multi-processor platforms. Streaming applications essentially continue their execution indefinitely. Therefore, one of the key properties of an SDFG is liveness, i.e., whether all parts of the SDFG can run infinitely often. Another elementary requirement is whether an implementation of an SDFG is feasible using a limited amount of memory. In this paper, we study two interpretations of this property, called boundedness and strict boundedness, that were either already introduced in the SDFG literature or studied for other models. A third and new definition is introduced, namely self-timed boundedness, which is very important to SDFGs, because self-timed execution results in the maximal throughput of an SDFG. Necessary and sufficient conditions for liveness in combination with all variants of boundedness are given, as well as algorithms for checking those conditions. As a by-product, we obtain an algorithm to compute the maximal achievable throughput of an SDFG that relaxes the requirement of strong connectedness in earlier work on throughput analysis.

1 Introduction

Synchronous Data Flow Graphs (SDFGs, see [12]), also known as weighted Marked Graphs in Petri-net theory, are used widely in modelling and analyzing data flow applications. They are often used for modelling DSP applications [3, 18] and for designing concurrent multimedia applications implemented on multi-processor systems-on-chip [16]. The model is suitable for realizing a system with predictable performance properties as several analysis techniques like throughput analysis exist [8].

An SDFG is a graph with actors as vertices and channels as edges. Actors represent basic parts of an application which need to be executed. Channels represent data dependencies between actors. Execution of an actor is designated by an actor firing. Each actor generates a fixed number of tokens when it fires. These are stored in the channels with unlimited capacities. An execution of an SDFG is a sequence of actor firings which respects data dependencies. The exact order of actor firings is not determined.

Consequently, several executions exist for an SDFG. Because of the usage of SDFGs for modelling streaming applications, only those SDFGs which have executions in which all actors are fired infinitely often are of interest. This property of SDFGs is called liveness. Furthermore, only executions that require a finite amount of storage for the channels are of interest. This paper formally studies three different interpretations of this second property, all in combination with liveness.

The paper investigates two known interpretations, namely boundedness (whether there exists a bounded execution of an SDFG) and strict boundedness (whether all executions are bounded). We prove necessary and sufficient conditions guaranteeing that an SDFG is live and (strictly) bounded. For strict boundedness, these conditions follow immediately from a similar result known for Petri nets.

The natural way of executing an SDFG in which all actors fire as soon as they can fire, is called self-timed execution. This execution is important since it leads to the maximal obtainable throughput of an SDFG [18]. Because of the importance of self-timed execution of SDFGs and its applications in the context of multi-processor systems, a new notion of boundedness, namely self-timed boundedness is introduced. This notion requires that self-timed execution of SDFGs is bounded. Necessary and sufficient conditions for the liveness and self-timed boundedness of SDFGs are proved. These conditions heavily depend on the throughput of actors (average number of firings of an actor per time unit). Existing techniques for throughput calcula-
2 Synchronous Data Flow Graphs

2.1 Basic Definitions

This section formally defines SDFGs and some of their basic properties. Let $\mathbb{N}_0 = \{0, 1, \ldots\}$ (and $\mathbb{N} = \mathbb{N}_0 \setminus \{0\}$) denote the (positive) natural numbers. The following definition captures the structure of an SDFG.

**Definition 1** [Synchronous Data Flow Graph (SDFG)] An SDFG is a pair $(A, C)$, where $A$ denotes the set of actors and $C \subseteq A^2 \times \mathbb{N}^2$ the set of channels. Each $(s, d, p, c) \in C$ denotes that actor $d$ depends on actor $s$, where $p$ and $c$ are the production and consumption rates of tokens of $s$ and $d$, respectively. The predecessors of $a$ in $\text{Pred}(a) = \{s \in A \mid (s, a, p, c) \in C\}$ are those actors on which $a$ depends. The channels between $a$ and its predecessors are referred to as the input channels of $a$, denoted by $\text{IC}(a)$. Similarly, the successors of $a$ in $\text{Succ}(a) = \{d \in A \mid (a, d, p, c) \in C\}$ are those actors that depend on $a$. The output channels (channels between $a$ and its successors) of $a$ are denoted by $\text{OC}(a)$. We call a channel from an actor $a$ to itself a self-loop channel. We denote the set of self-loop channels of an actor $a$ by $\text{SLC}(a) = \text{IC}(a) \cap \text{OC}(a)$. An SDFG in which all production and consumption rates are one is called a Homogeneous SDFG (HSDFG).

Figure 1 shows a simple example of an SDFG. Actors are labeled with their names and execution times (introduced later). Channels are labeled with production and consumption rates. The black dots are tokens. To capture the execution of an SDFG, we define the channel state of an SDFG as the distribution of tokens over its channels.

**Definition 2** [Channel State] A channel state of an SDFG $(A, C)$ is a function $S : C \rightarrow \mathbb{N}_0$ that returns the number of tokens stored in each channel. Each SDFG has an initial channel state $S_0$ denoting the number of tokens that are initially stored in the channels.

An execution of an SDFG is defined based on the firings of its actors, which may lead to changes in the channel state.

**Definition 3** [Firing] Let $a \in A$ be an actor of an SDFG $(A, C)$). Actor $a$ is said to be enabled in channel state $S$ in case $S(e) \geq c$ for all input channels $e = (s, a, p, c)$ in $\text{IC}(a)$. If $a$ is enabled in $S_0$ and it fires, the resulting channel state $S_{t+1}$ is defined by $S_{t+1}(e) = S_t(e) - c$ for each input channel $e = (s, a, p, c)$ in $\text{IC}(a) \setminus \text{SLC}(a)$, $S_{t+1}(e) = S_t(e) + p$ for each output channel $e = (a, d, p, c)$ in $\text{OC}(a) \setminus \text{SLC}(a)$, $S_{t+1}(e) = S_t(e) + p - c$ for each self-loop channel $e = (a, a, p, c) \in \text{SLC}(a)$, and $S_{t+1}(e) = S_t(e)$ for all channels $e \notin \text{IC}(a) \cup \text{OC}(a)$.

**Definition 4** [Execution and Maximal Execution] Let $S_0$ denote the initial channel state of an SDFG $(A, C)$. An execution $\sigma$ of $(A, C)$ is a (finite or infinite) sequence of channel states $S_0, S_1, \ldots$ such that $S_{t+1}$ is the result of firing an enabled actor in $S_t$ for all $i \geq 0$. An execution is maximal if and only if it is finite with no actors enabled in the final channel state, or if it is infinite.

Not all SDFGs are considered to be useful in practice. One normally seeks a system that is deadlock-free or live.

**Definition 5** [Deadlock and Liveness] An SDFG has a deadlock if and only if it has a maximal execution of finite length. An SDFG is live if and only if it has an execution in which all actors fire infinitely often.

It is known [10] that the execution of an SDFG is deterministic, which means that the order of execution does not affect the states that can eventually be reached. Thus, if one execution of an SDFG deadlocks, then all executions deadlock. The example SDFG $G_{ex}$ is live.

2.2 Timed SDFGs

For performance analysis of streaming applications, an SDFG is often extended with time.
that assigns to each actor the amount of time it takes to fire, where \( \mathbb{Q}_0^+ \cup \{\infty\} \) is the set of positive rational numbers plus 0 and \( \infty \). For \( a \in A \), \( E(a) \) is referred to as the execution time of \( a \).

**Definition 7** [Timed SDFG] A timed SDFG is a triple \( (A, C, E) \) denoting an SDFG \( (A, C) \) with execution time \( E \).

The infinite execution times are used lateron to model deadlocks. Normally, SDFGs do not have infinite actor execution times.

Notice that actor firings in a timed SDFG are not atomic. Firing an actor now takes time. To define the state of a timed SDFG, we assume that all changes in the number of tokens on all channels of an actor happen at the end of its firing.

**Definition 8** [Timed State] A state of a timed SDFG \( (A, C, E) \) is a pair \( (S, \tau) \), where \( S \) is a channel state and \( \tau \in \mathbb{Q}_0^+ \) is the accumulated time. The initial state \( (A, C, E) \) is given by the initial channel state \( S_0 \) and the start time of the system \( \tau_0 = 0 \).

We denote the number of completed firings of an actor \( a \in A \) which occurred up to time \( \tau \) by \( F_{a,\tau} \).

Among all timed executions there are some of special interest. A timed execution for which the firing of an actor always starts as soon as possible is called a self-timed execution. Self-timed executions are important in the context of performance analysis because they imply obtaining the maximal attainable throughput [18].

**Definition 9** [Self-timed Execution] A timed execution is called self-timed if and only if it is maximal and all actors start their firing as soon as they are enabled.

If two or more actors complete their firing at some point in time in a self-timed execution, the order of their appearance in the execution is not determined. In other words, any permutation of such actor firings results in a self-timed execution. Thus, the number of self-timed executions is larger than one in such cases. Note that in all self-timed executions the start and end times of firings of all actors are equal. Also the channel states after completion of all actor firings that can complete at a certain point in time are the same in all self–timed executions.

**Definition 10** [Self-timed Execution] A timed execution is called self-timed if and only if it is maximal and all actors start their firing as soon as they are enabled.

Figure 2 illustrates a self-timed execution of the example SDFG \( G_{ex^*} \) of Figure 1. The state contains a channel component with the distribution of tokens over the channels \( a-a, a-b, b-c, c-b \), respectively, and a time component. In the depicted cycle, the time component is denoted symbolically to emphasize that the behavior repeats itself every six time units, after some initial transient phase.

### 2.3 Structural Properties

The directed graph of an SDFG has some structural properties that are relevant for deciding boundedness. This paper assumes connected SDFGs for which the directed graph consists of one component. SDFGs consisting of multiple components can be considered as a set of single-component SDFGs, which can be analyzed separately.

A well known stronger form of connectivity is given by the following two definitions.

**Definition 11** [Path and Cycle] A directed path \( p \) is a sequence of actors \( a_1, a_2, \ldots, a_l \) such that \( a_{i+1} \in Succ(a_i) \) for all \( 1 \leq i < l \). Path \( p \) is simple iff \( a_i \neq a_j \) for all \( i \neq j \). If \( a_1 = a_l \) and \( l \geq 2 \), then \( p \) is said to be a cycle.

**Definition 12** [Strongly Connected SDFG] An SDFG is strongly connected iff there exists a directed path from any actor to any other actor. Any subgraph of an SDFG which is strongly connected is called a strongly connected component (SCC, for short). An SCC \( \kappa \) is maximal iff there is no SCC \( \kappa' \) where \( \kappa' \) is a strict subgraph of \( \kappa \).

Another structural property of SDFGs concerns the correspondence between production and consumption rates.

**Definition 13** [Consistency and Balance Equations] A repetition vector for an SDFG \( (A, C) \) is a function \( \gamma : A \to N_0 \) such that for every \( (s, d, p, c) \in C \), the equation \( p \gamma(s) = c \gamma(d) \) holds. These equations are called balance equations. Repetition vector \( \gamma \) is called non-trivial iff \( \gamma(a) > 0 \) for all \( a \in A \). If a non-trivial repetition vector exists, the SDFG is called consistent. The smallest non-trivial repetition vector of a consistent SDFG is referred to as the repetition vector.
Note that the definitions in this subsection carry over to timed SDFGs in a straightforward way. Timed SDFG $G_{ex}$ is consistent with repetition vector $(a \mapsto 3, b \mapsto 3, c \mapsto 2)$.

### 2.4 Throughput of Timed SDFGs

In this section the throughput of timed SDFGs is defined, and the relation between the execution of an SDFG and its throughput is explained.

**Definition 14 [Throughput]** The throughput $Th(a)$ of an actor $a$ for a self-timed execution of a timed SDFG $(A, C, E)$ is defined as the average number of firings of $a$ per time unit. Formally,

$$Th(a) = \lim_{\tau \to \infty} \frac{F_{a,\tau}}{\tau}.$$  

If $G = (A, C, E)$ is consistent, then its throughput is defined as

$$Th(G) = \min_{a \in A} \frac{Th(a)}{\gamma(a)},$$

where $\gamma$ is the repetition vector of $(A, C, E)$. That is, the throughput of $G$ is the minimal actor throughput normalized by the repetition vector.

We define the local throughput of an actor as the throughput of that actor in a self-timed execution where non-self-loop input channels are removed; in other words, the throughput of an actor when it does not need to wait for data from other actors.

**Definition 15 [Local Throughput]** The local throughput $LTh(a)$ of an actor $a$ for a self-timed execution of a timed SDFG $(A, C, E)$ is defined as

$$LTh(a) = \begin{cases} 0, & \text{if there is a } ch = (a, a, p, c) \in \text{SLC}(a) \text{ such that } p < c \text{ or } S_0(ch) < c \\ \min_{ch=(a, a, r, r) \in \text{SLC}(a)} \frac{S_0(ch)/r}{E(a)}, & \text{otherwise.} \end{cases}$$

If an actor has a self-loop channel with a lower production rate than consumption rate or insufficient tokens for an initial firing, its local throughput is zero, i.e., it deadlocks at some point in time. Otherwise, the local throughput is determined by the self-loop channels with equal production and consumption rates. If there are no such channels, i.e., there are no self-loop channels or all self-loop channels have a higher production than consumption rate, local throughput is by definition infinite.

In a self-timed execution of a timed SDFG, there is always a time $\tau_p$ after which only a repetitive pattern of actor firings occurs (when ignoring the order among actor firing completions occurring at the same moment in time) [8, 1]. The self-timed execution from the beginning up to time $\tau_p$ is called the transient phase, and thereafter is addressed as the periodic phase. Figure 2 illustrates this fact. Thus, the throughput of an arbitrary actor $a$ in the self-timed execution can be calculated by counting the number of occurrences of firings of $a$ in one period divided by the amount of time that the period takes. The firings of $a$ in one period can be spread over the period, but the number of firings of one actor in one period is always fixed.

Consider again SDFG $G_{ex}$ of Figure 1. The local throughput of actor $a$ is $\frac{1}{2}$, whereas it is $\infty$ for $b$ and $c$. The throughput of the three actors equals $\frac{2}{3} = \frac{1}{2}$, $\frac{2}{3} = \frac{1}{2}$, and $\frac{2}{3} = \frac{1}{2}$, respectively. The graph throughput $Th(G_{ex})$ is determined by actor $a$ (with repetition-vector entry 3) and is equal to $(\frac{2}{3})/3 = \frac{1}{4}$. This illustrates that the periodic behavior of the graph as a whole needs 6 time units per period.

### 2.5 Boundedness Definitions

Different useful notions of boundedness can be defined for SDFGs. To enable identifying these forms, we first define boundedness for a given execution.

**Definition 16 [Bounded Channel and Bounded Execution]** Let $\sigma = S_0, S_1, \ldots$ be an execution of an SDFG $(A, C)$. We call a channel $ch$ bounded under $\sigma$ iff there exists some $B \in \mathbb{N}$ such that $S_i(ch) \leq B$ for all $i \geq 0$. If all channels of the SDFG are bounded under $\sigma$ then $\sigma$ is bounded.

Definition 16 carries over to timed executions in a straightforward way. Now, we give a definition for the boundedness of an SDFG which intuitively means that it can be implemented using a finite amount of memory.

**Definition 17 [Bounded SDFG]** A (timed) SDFG is called bounded iff there exists a bounded maximal execution. It is unbounded otherwise.

A stronger form of boundedness is strict boundedness.

**Definition 18 [Strictly Bounded Channel and Strictly Bounded SDFG]** A channel is strictly bounded iff it is bounded under all executions. A (timed) SDFG is called strictly bounded iff all of its channels are strictly bounded.

Note that any strictly bounded SDFG is also bounded. We finally define another form of boundedness, which only considers self-timed executions of timed SDFGs.

**Definition 19 [Self-timed Bounded SDFG]** A timed SDFG is self-timed bounded iff all self-timed executions are bounded. A channel in a timed SDFG is self-timed bounded iff it is bounded in all self-timed executions.
All self-timed bounded SDFGs are bounded but not necessarily strictly bounded. Running example $G_{ex}$ is not strictly bounded because $a$ can be fired indefinitely without firing $b$ and $c$. However, it is self-timed bounded, as Figure 2 illustrates. It is not difficult to construct bounded SDFGs that are not self-timed bounded. If the execution times of actors $b$ and $c$ in $G_{ex}$ are changed to 3, for example, then the SDFG remains bounded but it is no longer self-timed bounded. These examples show that the notion of self-timed boundedness does not coincide with other notions of boundedness. Given the importance of self-timed execution, it is worth investigating this notion in some detail.

3 Boundedness

In this section, we study necessary and sufficient conditions under which an SDFG is live and bounded.

**Theorem 20** A live SDFG $G = (A, C)$ is bounded iff it is consistent.

**Proof** For an execution $\sigma = S_0, S_1, \ldots$, the $n$-th execution step refers to the fi ring performed just before entering channel state $S_n$. We use $F_{a,n}$ to denote the number of fi rings of a specifi c actor $a$ after performing $n$ execution steps.

The suffi cient part: If the graph is consistent, then there exists a non-trivial repetition vector $\gamma$ for $G$. So, starting from the initial channel state $S_0$, if every actor $a \in A$ fi res $\gamma(a)$ times, then according to the defi nition of the repetition vector the channel state of $G$ goes back to $S_0$. This procedure is always possible if $G$ is live [11]. As the number of initial tokens, the number of fi rings and the rates are bounded, therefore the number of produced tokens is limited. So, we conclude that the required memory under these fi rings is bounded. Therefore, the execution consisting of repeating the same actor fi ring pattern is bounded.

The necessary part: If $G$ is live and bounded, then there exists an infi nite execution $\sigma$ which is bounded, and thus visits some channel states repeatedly. Suppose that both the $n$-th and $n'$-th element of $\sigma$ are channel state $S$. We can calculate the number of tokens on every channel $c$ at step $k$ by the following expression

$$S_k(ch) = S_0(ch) + pF_{a,k} - cF_{b,k}.$$ 

Assume without loss of generality that $n' > n$. Since $G$ is connected, it is impossible to return to the same state without having fi red every actor at least once. Therefore, $F_{a,n'} > F_{a,n}$ and $F_{b,n'} > F_{b,n}$. Since we have $S_n = S_{n'}$, it follows that

$$(F_{a,n'} - F_{a,n})p = (F_{b,n'} - F_{b,n})c.$$

Hence, if we take for all $a \in A$, $\gamma(a) = F_{a,n'} - F_{a,n}$ then $\gamma$ is a non-trivial solution for the balance equations, which means $G$ is consistent.

Theorem 20 states the consistency of an SDFG as a necessary and sufficient condition for boundedness of live SDFGs. If a subgraph of an SDFG deadlocks (which means that the SDFG is not live) then the consistency of an SDFG is not sufficient for boundedness. For example, consider $G_{ex}$ of Figure 1 without the initial token in the $c-b$ channel. Execution times may be ignored. The resulting SDFG is consistent but not bounded. The SCC of the graph that consists of actors $b$ and $c$ deadlocks after the fi rst fi ring of both actors. However, actor $a$ can continue its fi ring, which leads to an unbounded channel between $a$ and $b$.

**Proposition 21** [19] A strongly connected SDFG is live iff it is deadlock-free.

The defi nition of liveness states that a live SDFG has an execution in which all actors fi re infi nitely often. If a live SDFG is strongly connected, then all actors fi re infi nitely often in all maximal executions.

**Lemma 22** If one SCC in an SDFG $G$ deadlocks then either $G$ deadlocks or it is unbounded.

**Proof** If $G$ contains only one SCC then the lemma follows immediately. In case it consists of multiple SCCs, at least one deadlocked and at least one deadlock-free, then there exists an SCC (possibly a single actor) which is deadlock-free and connected to an actor in a deadlocked SCC. This connecting channel must necessarily go from the deadlock-free SCC to the deadlocked SCC. Since in a deadlock-free SCC all actors must necessarily fi re infi nitely often, this channel must be unbounded.

This lemma implies that a deadlock-free and bounded SDFG is live.

**Corollary 23** An SDFG is live and bounded iff it is deadlock-free and bounded.

The following theorem follows from Theorem 20, Proposition 21, Lemma 22, and Corollary 23.

**Theorem 24** An SDFG is live and bounded iff it is consistent and all its SCCs are deadlock-free.

**Proof** For the necessary part, note that Theorem 20 and the fact that an SDFG is live and bounded implies that it is consistent. Liveness and boundedness together with Lemma 22 shows that all SCCs must be deadlock-free. For the suffi cient part, observe that the fact that all SCCs are deadlock-free implies liveness of the SDFG, because the SCCs of the SDFG without input channels from other SCCs can, by Proposition 21, always continue feeding tokens into the SDFG, which again by Proposition 21 implies liveness of all SCCs and hence the SDFG. Theorem 20 then implies that the SDFG is also bounded.

The example SDFG $G_{ex}$ is live and bounded because it is consistent and all its SCCs are deadlock-free.

Next, we give an algorithm to check liveness and boundedness of an SDFG.
**Algorithm isLive&Bounded(G)**

**Input:** A connected (timed) SDFG $G$

**Output:** “live and bounded” or “either deadlock or unbounded”

1. if $G$ is inconsistent
2. then return “either deadlock or unbounded”
3. for each maximal SCC $S$ in $G$
4. do if $S$ deadlocks
5. then return “either deadlock or unbounded”
6. return “live and bounded”

Consistency of SDFGs can be verified efficiently as explained in [3]. Maximal SCCs of a graph can also be computed efficiently [5]. Algorithms for detecting deadlock for consistent strongly connected SDFGs that are efficient in practice are given in [11, 8].

4 Strict Boundedness

This section identifies sufficient and necessary conditions for the liveness and strict boundedness of an SDFG.

**Theorem 25** [19, Theorem 4.11] A live (timed) SDFG is strictly bounded iff it is consistent and strongly connected.

This theorem in combination with Proposition 21 implies the following theorem.

**Theorem 26** An SDFG is live and strictly bounded iff it is deadlock-free, consistent and strongly connected.

So the algorithm for checking liveness and strict boundedness first checks whether the SDFG is strongly connected and consistent, and then whether it is deadlock-free using the algorithms from [5, 3, 8, 11]. The example of Figure 1 is not strictly bounded because it is not strongly connected.

5 Self-timed Boundedness

In this section, we investigate the liveness and self-timed boundedness of timed SDFGs. A self-timed execution of a live and self-timed bounded SDFG uses a finite amount of memory and all actors fire infinitely often in such an execution. Necessary and sufficient conditions are given, and an algorithm for checking these conditions.

5.1 Some Basic Properties

Self-timed boundedness has a strong relationship with the throughput of an SDFG. In this subsection, some properties for the throughput as well as the relation between boundedness and throughput of timed SDFGs are given.

The throughput of an actor is only determined by the throughput of its predecessors and its local throughput.

**Lemma 27** The throughput of an actor $b \in A$ of a timed SDFG $G = (A, C, E)$ satisfies the equation

$$Th(b) = \min\{ \min_{(a,b,p,c) \in IC(b)} \frac{p}{c} Th(a), LTh(b) \}.$$  \hfill (1)

**Proof** It is not difficult to see that the local throughput of an actor is an upper bound for its throughput. So, we prove the theorem for the case $\min_{(a,b,p,c) \in IC(b)} \frac{p}{c} Th(a) \leq LTh(b)$. Let channels $ch_i = (a_i, b, p_i, c_i)$ be all input channels of $b$. Suppose $a_m$ is an actor for which $(p_m/c_m) Th(a_m)$ is minimal.

First, consider the case that $c_m Th(b) - p_m Th(a_m) = k > 0$. By substituting the defi nition of actor throughput we get

$$\lim_{\tau \to \infty} \frac{c_m F_{b,\tau} - p_m F_{a_m,\tau}}{\tau} = k.$$  

According to the defi nition of a limit, for any $\epsilon > 0$, there exists a time $T$ where for all $\tau > T$, we have

$$\left| \frac{c_m F_{b,\tau} - p_m F_{a_m,\tau}}{\tau} - k \right| < \epsilon.$$  

If we set $\epsilon$ to $k/2$, we can conclude that for $\tau > T$,

$$c_m F_{b,\tau} - p_m F_{a_m,\tau} > \frac{k}{2} \tau,$$

which means that regardless of the number of initial tokens on the channel, there exists a $T$, such that for all $\tau > T$ the number of produced tokens is less than the consumed ones, which is impossible. Hence, $c_m Th(b) - p_m Th(a_m) \leq 0$.

Second, similarly, we can show that if $c_m Th(b) - p_m Th(a_m) < 0$, then there exists a time $T$, where for times $\tau > T$, there were enough tokens on $ch_m$ to fire and due to the choice for $a_m$ (minimality of $(p_m/c_m) Th(a_m)$) on all other input channels of $b$, while $b$ did not start a new ring, which contradicts the self-timed execution scheme. Thus, $c_m Th(b) - p_m Th(a_m) = 0$, which completes the proof.

The throughput of actor $b$ of $G_{ext}$, for example, is $\frac{3}{2}$, because its predecessor $a$ has that throughput, the rates of channel $a\rightarrow b$ are 1 and its local throughput is $\infty$.

**Corollary 28** If actors $a, b \in A$ of an SDFG $G$ are connected by a channel $(a, b, p, c)$ then $Th(b) \leq (p/c) Th(a)$.

After having illustrated the factors that are involved in calculating the throughput of an actor, we now show that the only case that a channel is not self-timed bounded, is when the production of tokens into one channel is larger than the consumption of tokens out of that channel.

**Lemma 29** SDFG $(A, C, E)$ is self-timed bounded iff $Th(b) \geq (p/c) Th(a)$ for every channel $(a, b, p, c) \in C$.

**Proof** We know that there is a time $\tau_p$ such that for all $\tau \geq \tau_p$, the execution of $G$ is in the periodic phase. Let $d$ be the amount of
time that one period of a self-timed execution $\sigma$ of $G$ takes. Then for any actor $a \in A$ and time $\tau \geq \tau_p$ we have

$$F_{a, \tau + d} - F_{a, \tau} = d Th(a).$$

Although the number of fi rings of one actor in one period is always fi xed, the fi rings of an actor in one period can be spread over the period. Therefore, we have the following inequality, where $k = \lfloor (\tau - \tau_p)/d \rfloor$,

$$kd Th(a) \leq F_{a, \tau} - F_{a, \tau_p} \leq (k + 1)d Th(a). \quad (2)$$

Let $S_r(ch)$ be the number of tokens on channel $ch = (a, b, p, c) \in C$ at time $\tau$, then $S_r(ch)$ can be calculated by

$$S_r(ch) = S_0(ch) + p F_{a, \tau} - c F_{b, \tau}.$$

(3)

For $\tau > \tau_p$, by using Equation (2), we have

$$S_r(ch) \leq S_0(ch) + p (F_{a, \tau} + (k + 1)d Th(a)) - c (F_{b, \tau} + kd Th(b)).$$

Since the only part of the above inequality that depends on $k$ is $kd(p Th(a) - c Th(b))$, $S_r(ch)$ is bounded if $p Th(a) \leq c Th(b)$. Also we have

$$S_r(ch) \geq S_0(ch) + p (F_{a, \tau} + kd Th(a)) - c (F_{b, \tau} + (k + 1)d Th(b)).$$

Using the same argument as above we can conclude that $ch$ is not bounded if $p Th(a) > c Th(b)$. \qed

The next proposition gives necessary and sufcient conditions for self-timed boundedness of a live strongly connected SDFG.

**Proposition 30** A live and strongly connected SDFG $G$ is self-timed bounded if and only if it is consistent.

**Proof** The suficient part can be deduced directly from Theorem 25 as strict boundedness ensures self-timed boundedness. For the necessary part, the same argument as used in the proof of Theorem 20 is valid. \qed

Lemmas 31 and 32 and Proposition 33 prove some useful properties about the relation between the throughput of two arbitrary actors in consistent self-timed bounded or strongly connected SDFGs. Lemma 31 follows immediately from Corollary 28 and Lemma 29. Proposition 32 shows the relation between producer and consumer actors of an arbitrary self-timed bounded channel. Lemma 32 shows the relation between the actor throughputs for any two actors in an SCC of an SDFG. Proposition 33 gives the relation between the throughput of two arbitrary actors in consistent self-timed bounded or strongly connected SDFGs.

**Lemma 31** If a channel $(a, b, p, c)$ connecting $a$ and $b$ is self-timed bounded then $Th(b) = (p/c) Th(a)$.

**Lemma 32** If $a$ and $b$ are two actors of an SCC of a consistent SDFG with repetition vector $\gamma$, then $Th(a)/\gamma(a) = Th(b)/\gamma(b)$.

**Proof** We know that actors $a$ and $b$ are on a cycle. Let $a = i_1, i_2, \ldots, i_k = b$, then $\alpha$ denotes this cycle. If actors $i_1$ and $i_2$ are connected by channel $(i_1, i_2, p_1, c_1)$, and $i_2$ and $i_3$ by $(i_2, i_3, p_2, c_2)$ and so forth, then, by Corollary 28, we know that

$$Th(a) = \frac{p_1}{c_1} Th(i_1),$$

$$Th(i_i - 1) = \frac{p_{i - 2}}{c_{i - 2}} Th(i_{i - 2}),$$

$$\cdots \leq$$

$$Th(i_{k - 1}) = \frac{p_k}{c_k} Th(b),$$

$$Th(b) = \frac{p_{k - 1}}{c_k} Th(i_{k - 1}),$$

$$\cdots \leq$$

$$Th(i_2) = \frac{p_1}{c_1} Th(a).$$

(4)

By combining the above equations, we obtain

$$Th(a) \leq \frac{p_{i - 2} p_{i - 3} \cdots p_k}{c_{i - 2} c_{i - 3} \cdots c_k} Th(b),$$

and

$$Th(b) \leq \frac{p_k p_{k - 1} \cdots p_1}{c_k c_{k - 1} \cdots c_1} Th(a).$$

(5)

According to the defnition of $\gamma$ we can also write

$$c_{i - 1} \gamma(a) = p_{i - 1} \gamma(i_{i - 1}) \Rightarrow \frac{p_{i - 1}}{c_{i - 1}} = \frac{\gamma(a)}{\gamma(i_{i - 1})},$$

$$c_{i - 2} \gamma(i_{i - 1}) = p_{i - 2} \gamma(i_{i - 2}) \Rightarrow \frac{p_{i - 2}}{c_{i - 2}} = \frac{\gamma(i_{i - 1})}{\gamma(i_{i - 2})},$$

$$\cdots \leq$$

$$c_{k - 1} \gamma(i_{k - 1}) = p_k \gamma(b) \Rightarrow \frac{p_k}{c_k} = \frac{\gamma(i_{k - 1})}{\gamma(b)},$$

$$c_k \gamma(b) = p_{k - 1} \gamma(i_k) \Rightarrow \frac{p_{k - 1}}{c_k} = \frac{\gamma(b)}{\gamma(i_k)},$$

$$\cdots \leq$$

$$c_1 \gamma(i_2) = p_1 \gamma(a) \Rightarrow \frac{p_1}{c_1} = \frac{\gamma(i_2)}{\gamma(a)}.$$

By substitution into Inequalities (4) and (5) we have $Th(a) \leq (\gamma(a)/\gamma(b)) Th(b)$ and $Th(b) \leq (\gamma(b)/\gamma(a)) Th(a)$. Rewriting this result yields $Th(a)/\gamma(a) \leq Th(b)/\gamma(b)$ and $Th(b)/\gamma(b) \leq Th(a)/\gamma(a)$, which means that $Th(a)/\gamma(a) = Th(b)/\gamma(b)$, completing the proof. \qed

**Proposition 33** If $a$ and $b$ are two actors of a consistent self-timed bounded or strongly connected SDFG $G$ with repetition vector $\gamma$ then $Th(a)/\gamma(a) = Th(b)/\gamma(b)$.

**Proof** The desired result for strongly connected SDFGs follows immediately from Lemma 32. Therefore, assume that $a$ and $b$ are actors of a consistent self-timed bounded SDFG. Suppose path $a = i_1, i_2, \ldots, i_k = b$ is a path connecting $a$ to $b$. If actors $i_1$
and \( i_2 \) are connected by \((i_1, i_2, p_1, c_1)\) and \( i_2, i_3 \) are connected by \((i_2, i_3, p_2, c_2)\) and so on, then according to Lemma 31 we have

\[
Th(b) = \frac{p_k}{c_k-1} Th(i_{k-1}),
\]

\[
Th(i_{k-1}) = \frac{p_{k-2}}{c_{k-2}} Th(i_{k-2}),
\]

\[
\cdots \cdots
\]

\[
Th(i_2) = \frac{p_1}{c_1} Th(a).
\]

The proof can now be completed along the lines of the proof of Lemma 32.

This proposition shows that for consistent self-timed bounded or strongly connected SDFGs the throughput as defined in Definition 14 can be calculated via an arbitrary actor without explicitly computing the minimum.

5.2 Reduction to an HSDFG

In this section we propose a method for reducing a consistent SDFG \( G \) to an HSDFG \( G_H \) which preserves (non-)liveliness and self-timed (un-)boundedness of \( G \). In \( G_H \), every actor has a self-loop channel with one initial token, rates of all channels are one (i.e., it is an HSDFG), and, ignoring self-loops, it is acyclic.

Because of these simple properties, we use the reduced graph for verifying the liveliness and self-timed boundedness of the original SDFG. The reduction also preserves throughput which means our algorithm also provides the throughput of the original SDFG \( G \).

The reduction uses the notion of local throughput of an SCC of an SDFG, and it is illustrated in Figure 3 which provides the reduced graph for the running example.

**Definition 34** [Local Throughput of an SCC] The local throughput \( LTh(\kappa) \) of an SCC \( \kappa = (A_\kappa, C_\kappa, E_\kappa) \) in a consistent SDFG \( G = (A, C, E) \) with repetition vector \( \gamma \) is defined as the local throughput of an arbitrary actor \( a \in A_\kappa \) when all input channels from \( A \setminus A_\kappa \) to \( A_\kappa \) are removed, divided by \( \gamma(a) \).

Lemma 32 implies that this definition is sound.

![Figure 3. The reduced HSDFG for \( G_{ex} \).](image)

**Definition 35** [Reduced Graph] Let a consistent SDFG \( G = (A, C, E) \) contain \( n \) maximal SCCs \( \kappa_1 = (A_{\kappa_1}, C_{\kappa_1}, E_{\kappa_1}), \ldots, \kappa_n = (A_{\kappa_n}, C_{\kappa_n}, E_{\kappa_n}) \). Suppose \( \gamma \) is the repetition vector of \( G \). We define the reduced SDFG \( G_H = (A_H, C_H, E_H) \) as follows: \( A_H = \{ x_i | 1 \leq i \leq n \} \) (which means one actor for each maximal SCC in \( G \)); \( C_H \) contains a channel \((x_i, x_j, p \kappa(a), c \kappa(b))\) for every channel \((a, b, p, c) \in C \) where \( a \in A_{\kappa_i}, b \in A_{\kappa_j}, i \neq j \); \( C_H \) also contains self-loop channels \((x_i, x_i, 1, 1)\) for every actor; the execution time \( E_H(x_i) \) equals \( 1 / LTh(\kappa_i) \) if \( \kappa_i \) does not deadlock and \( \infty \) if it does. According to the balance equations we know that for each channel in the original graph \((a, b, p, c), f \kappa(a) = c \kappa(b)\). Thus, the production and consumption rates for every channel in \( C_H \) are equal. Therefore, we can simplify the reduced \( G \) by setting all rates of all channels in \( C_H \) to one. Consequently, we obtain an HSDFG as the result. Finally, every self-loop channel in \( G_H \) contains one initial token, and all the other channels are empty.

Since the HSDFG resulting from the reduction is acyclic when ignoring self-loops, the preservation of throughput, (non-)liveliness and self-timed (un-)boundedness that we are aiming at, is independent of the number of initial tokens on the non-self-loop channels. Hence, we choose to leave those channels empty.

Consider the reduced graph shown in Figure 3. The original graph \( G_{ex} \) has two maximal strongly connected components, containing actor \( a \), and actors \( b \) and \( c \), respectively. These SCCs are reduced to actors \( x_1 \) and \( x_2 \). Since actor \( a \) has throughput \( \frac{1}{6} \) and repetition-vector entry \( 3 \), the execution time of \( x_1 \) is set to 6, illustrating that 3 firings of \( a \) take 6 time units. Considering the other SCC in isolation, it can be verified that one period of this SCC containing 3 firings of \( b \) and 2 of \( c \) consists of 4 time units. Given the repetition vector of \( G_{ex} \) and Definition 34, this gives a local throughput of \( \frac{1}{4} \) and an execution time of 4 for \( x_2 \).

The following proposition shows the relation between the throughput of actors in a maximal SCC of an SDFG and the throughput of the actor corresponding to that SCC in the reduced SDFG.

**Proposition 36** Let \( G_H \) be the reduced SDFG of a consistent timed SDFG \( G \) with repetition vector \( \gamma \). If a maximal SCC \( \kappa = (A_\kappa, C_\kappa, E_\kappa) \) in \( G \) is replaced by actor \( x \) in \( G_H \), then for every \( a \in A_\kappa \), \( Th(a) = \gamma(a) Th(x) \).

**Proof** For proving this proposition, first we define an intermediate reduced graph in which only one of the SCCs is reduced. Then, we prove the proposition for this intermediate SDFG. Finally, the result is proven for the entire reduction.

Let \( \kappa = (A_\kappa, C_\kappa, E_\kappa) \) be a maximal SCC in \( G \). \( \gamma \) be the repetition vector of \( G \) and \( LTh(\kappa) \) be the local throughput of \( \kappa \). For any fresh actor name \( x \notin A \), we define the \( k \)-reduced SDFG \( G_{\kappa-x} = (A_\kappa, C_\kappa, E_\kappa) \) as follows: \( A_x = (A_\kappa \setminus \{ x \}) \cup \{ x \} \); \( C_x \) equals \( C \setminus C_\kappa \), with every input channel \((a, k, p, c)\) with \( a \in A_\kappa \) and \( k \in A_\kappa \) replaced by \((a, x, p, c \gamma(k))\), every output channel \((k, a, p, c)\) with \( a \in A_\kappa \), and \( k \in A_\kappa \) replaced by \((x, a, p \gamma(k), c)\), and an extra self-loop channel \((x, x, 1, 1)\) with one initial token. The execution time \( E_x(a) \) equals \( E(a) \) for all actors \( a \in A \setminus A_\kappa \), and for actor

\[ G_H = (A_H, C_H, E_H) \]
it is set to $1/LTh(\kappa)$ if $\kappa$ does not deadlock and $\infty$ if it does. The channels of $G_{n\rightarrow x}$ contain the same number of initial tokens as the corresponding channels in $G$.

Next, for $G_{n\rightarrow x}$, we prove the following equation, for every $a \in A_n$.

$$Th(a) = \gamma(a) Th(x). \quad (6)$$

First, assume $\kappa$ has only self-timed unbounded input channels, if it has any input channels at all. Since $\kappa$ is a maximal SCC, the actor $f_i$ rings in $\kappa$ do not have impact on the production of any tokens in the input channels of $\kappa$. By the construction of $G_{n\rightarrow x}$, any part of the graph producing tokens into input channels of $\kappa$ remains unchanged in $G_{n\rightarrow x}$. Furthermore, because all input channels of $\kappa$ are self-timed unbounded, $\kappa$ is not constrained by its input channels realizing a throughput equal to $LTh(\kappa)$, and because $Th(x) \leq LTh(x) = LTh(\kappa)$. $x$ consumes tokens in a self-timed execution at most as fast as $\kappa$ consumes the corresponding tokens. Therefore, also all input channels of $x$ in $G_{n\rightarrow x}$ are self-timed unbounded. This means that at some point in time $x$ never has to wait for input tokens, which implies that $Th(x) = LTh(x)$. By the def of the local throughput of an SCC, $LTh(x) = LTh(\kappa) = Th(a)/\gamma(a)$, for some arbitrary $a$ in $A_n$, which proves Equation (6).

Second, we may assume that not all input channels of $\kappa$ are unbounded. By Lemma 32, it suffices to prove Equation (6) for an arbitrary actor of $\kappa$.

Let channels $ch_i = (b_i, a_i, p_i, c_i)$ be all channels connecting some actor $b_i$ in $A \setminus A_n$ to an actor $a_i$ in $A_n$. Denote this set as $IC(\kappa)$. Based on Corollary 28, $Th(\alpha_i) \leq (p_i/c_i) Th(b_i)$ for all $i$. The def of the local throughput of an SCC means that $Th(\alpha) = Th(a)/\gamma(a)$, for some arbitrary $a$ in $A_n$, which proves Equation (6).

Next, we prove Equation (6) for an arbitrary actor of $\kappa$.

Let $ch_m = (b_m, a_m, p_m, c_m)$ be a channel from actor $b_m \in A \setminus A_n$ to actor $a_m \in A_n$, such that, for all $i$,

$$Th(x) \leq \frac{p_i}{c_i \gamma(a_i)} Th(b_i).$$

i.e., $ch_m$ is a channel which constrains the throughput of $x$ the most. We continue to prove Equation (6) for actor $a_m$, i.e., we prove that $Th(a_m) = \gamma(a_m) Th(x)$.

We show that $ch_m$ is bounded. To show this by contradiction we assume that $ch_m$ is unbounded. According to Lemma 29 we have

$$p_m Th(b_m) > c_m Th(a_m) \Rightarrow \frac{p_m}{c_m \gamma(a_m)} Th(b_m) > \frac{Th(a_m)}{\gamma(a_m)}.$$

Since we assumed that not all $ch$ are unbounded, there exists a bounded channel $ch_k \in IC(\kappa)$. Therefore using first Lemma 32 and then Lemma 31 we can conclude that

$$\frac{p_m}{c_m \gamma(a_m)} Th(b_m) > \frac{Th(a_m)}{\gamma(a_m)} = \frac{p_k}{c_k \gamma(a_k)} Th(b_k),$$

which means

$$\frac{p_m}{c_m \gamma(a_m)} Th(b_m) > \frac{p_k}{c_k \gamma(a_k)} Th(b_k). \quad (7)$$

which contradicts the choice of $ch_m$. Hence, channel $ch_m$ must be bounded.

According to Lemma 27, the throughput of $x$ can be calculated as follows:

$$Th(x) = \min \left\{ \frac{p_i}{c_i \gamma(a_i)} Th(b_i), LTh(\kappa) \right\}.$$

The choice for $ch_m$ implies that we can calculate the throughput of $x$ as

$$Th(x) = \min \left\{ \frac{p_m}{c_m \gamma(a_m)} Th(b_m), LTh(\kappa) \right\}.$$

If the result of this minimum is $LTh(\kappa)$ then from the def of the through of an SCC and the def of the execution time of $x$, Equation (6) follows for $a_m$. In the other case, as $(b_m, a_m, p_m, c_m)$ is bounded, using Lemma 31, we have

$$Th(x) = \frac{p_m}{c_m \gamma(a_m)} Th(b_m) = \frac{Th(a_m)}{\gamma(a_m)},$$

which completes the proof for actor $a_m$, and hence for all actors in $A_n$.

Finally, it is not difficult to see that by replacing all SCCs of SDFG $G$ via the above intermediate reduction results in the reduced SDFG as def and in Defn 35 before all rates are replaced by ones. Therefore, we can extend Equation (6) to all SCCs of $G$, since the simplification of the rates in the reduction does not change the repetition vector, and therefore also the throughput does not change.

It is easy to verify that Proposition 36 holds for the running example. Consider for instance actor $x_2$ of the reduced graph. Its throughput in the reduced graph is fully determined by the throughput of $x_1$ and becomes therefore $\frac{1}{2}$. Proposition 36 states that $Th(b) = 3(\frac{1}{2}) = \frac{1}{2}$ and $Th(c) = 2(\frac{1}{2}) = 1$, which corresponds to the throughput values for $b$ and $c$ computed at the end of Section 2.4.

The next corollary follows from the definition of throughput, the observation that all repetition-vector entries of an HSDFG are always one, and Propositions 33 and 36.

**Corollary 37** The throughput of a consistent self-timed bounded SDFG is equal to the throughput of its reduced graph.

The reduction also preserves self-timed (un-)boundedness.

**Theorem 38** A consistent timed SDFG is self-timed bounded iff its reduced graph is self-timed bounded.

**Proof** In this proof, we again use the intermediate reduced graph as explained in the proof of Proposition 36, in which only one of the SCCs is replaced by an actor in the intermediate reduced SDFG. Let an SCC $\kappa = (A_n, C_n, E_n)$ of $G$ be replaced by an actor $x$ in $G_{n\rightarrow x}$. According to Theorem 25, channels in a live and strongly connected SDFG are strictly bounded, which implies the self-timed boundedness of channels in $C_n$. We want to prove that the input and output channels of actor $x$ in the reduced graph are self-timed bounded if the corresponding channels to/from $\kappa$ are self-timed bounded. First, we prove the sufficient part. If input channel $(b, a, p, c)$ in $G$
connecting actor \( b \in A \setminus A_e \) to \( a \in A_e \) is self-timed bounded, by Lemma 29 and Proposition 36 we have

\[
pTh(b) \leq cTh(a) = c\gamma(a)Th(x).
\]

Thus, by Lemma 29 channel \((b, x, p, c\gamma(a))\) is also self-timed bounded.

Now, we prove the sufficient condition for output channels from \( k \) in the same way. Suppose channel \((a, b, p, c)\) connecting an actor \( a \) in \( A \setminus A_e \) to an actor inside \( A_e \) is not self-timed bounded. We prove that channel \((x, b, p, c\gamma(a), c)\) is also self-timed bounded. Again using Lemma 29 and Proposition 36 we have

\[
pTh(a) \leq cTh(b) \Rightarrow p\gamma(a)Th(x) \leq cTh(b),
\]

which by Lemma 29 proves self-timed boundedness of \((x, b, p\gamma(a), c)\).

Next, we prove the necessary part of the proof, namely, if a channel \((b, a, p, c)\) connecting an actor \( b \in A \setminus A_e \) to an actor inside \( A_e \), is not self-timed bounded, then the channel \((b, x, p, c\gamma(a))\) is also not self-timed bounded.

Again in a similar way by replacing \( Th(a)\) with \( \gamma(a)Th(x) \) in the following inequality we can prove this part of the theorem.

\[
pTh(b) > cTh(a) = c\gamma(a)Th(x).
\]

Also for the case of an output channel \((a, b, p, c)\).

\[
pTh(a) > cTh(b) \Rightarrow p\gamma(a)Th(x) > cTh(b).
\]

Thus, we may conclude that the reduction of one SCC to a single actor preserves self-timed (un-)boundedness.

Applying the intermediate reduction on \( G \) can be done iteratively in different and independent steps. Therefore, applying the intermediate reduction for each SCC of \( G \) results in the reduced SDFG as defined by Definition 35 when after all the intermediate reductions all rates are changed to one. As proven in this theorem, each intermediate reduction preserves self-timed (un)boundedness. The change in rates does not influence this result because it affects production and consumption rates of each channel in the same way. Consequently, we can conclude that the reduced SDFG preserves self-timed (un)boundedness. \( \square \)

Proposition 36 implies that non-zero throughput (i.e., (non-)liveness) is preserved.

**Corollary 39** A consistent timed SDFG is live iff its reduced graph is live.

### 5.3 Verifying Self-timed Boundedness

This section introduces an algorithm that determines whether an SDFG is live and self-timed bounded. The following theorem follows from the results obtained so far.

**Theorem 40** A timed SDFG \( G \) is live and self-timed bounded iff is\( \text{Live} \& \text{SelftimedBounded}(G) \) returns “yes”.

**Algorithm isLive\&SelftimedBounded(G=(A, C, E))**

**Input:** A connected timed SDFG \( G \)

**Output:** “yes, \( Th(G) \)” if self-timed bounded and live, “no” otherwise

1. if not is\( \text{Live\&Bounded}(G) \) then return “no”
2. \( G_H = (A_H, C_H, E_H) \) ← reduce(\( G \))
3. \( AL[1..|A_H|] \) ← topologicalSort(\( G_H \))
4. if \( |A_H| = 1 \) then return “yes, \( \frac{1}{E_H(AL[1])} \)”
5. for \( i \leftarrow 1 \) to \( |A_H| \)
6. \( AL[i], Th = \frac{1}{E_H(AL[i])} \)
7. if \( \text{Pred}(AL[i]) = \{AL[i]\} \) and \( AL[i], Th = \infty \) then return “no”
8. max\( PTh = 0 \)
9. for each \( j \in \text{Pred}(AL[i]) \setminus \{AL[i]\} \)
10. \( do AL[i], Th \leftarrow \min(AL[i], Th, AL[j], Th) \)
11. max\( PTh \leftarrow \max(max\( PTh, AL[j], Th) \)
12. if max\( PTh \geq AL[i], Th \) then return “no”
13. return “yes, \( AL[1], Th \)”

The algorithm works in two steps. The first step checks the liveness and boundedness (as defined by Definition 17) of the graph by calling algorithm is\( \text{Live\&Bounded} \) (lines 1 and 2). If the graph is not live and bounded, it cannot be live and self-timed bounded. The second step concerns determining whether the reduced HSDFG is self-timed bounded (lines 3 to 17).

If is\( \text{Live\&Bounded} \) returns “yes”, we know that the SDFG is consistent. Then, line 3 of the algorithm reduces the SDFG according to Definition 35 and stores the result in \( G_H \). Note that the reduction requires throughput calculations for all SCCs. For efficiency reasons, these throughput calculations can be delayed till the algorithm really needs this information. Calculations may then be avoided if the algorithm returns “no” early. We have not made this explicit in the algorithm. Since \( G \) is at this point known to be live and consistent, by Corollary 39, also \( G_H \) is live. It remains to determine self-timed (un-)boundedness.

Ignoring self-loops, \( G_H \) is acyclic. Line 4 topologically sorts the actors of \( G_H \), and stores them in array \( AL \), so that the predecessors of an actor \( AL[i] \) are only among the \( AL[j] \) for \( j \leq i \). If \( G_H \) contains only one actor, then \( G \) is strongly connected, and hence, by Proposition 30, self-timed bounded, and the algorithm terminates. Based on Corollary 37, it returns the local throughput of the only actor of \( G_H \) as the throughput of \( G \). Note that every actor in a reduced graph has a self-loop channel with one token on it, so this value is equal to \( 1/E_H(AL[1]) \). Also note that \( E_H(AL[1]) \) and \( E_H(AL[i]) \) in general, may be 0. In this case, we assume that \( 1/E_H(AL[1]) \) is equal to \( \infty \).

Each iteration of the loop of lines 7 to 16 starts by calculating the local throughput of each actor \( AL[i], 1 \leq i \leq |A_H| \), storing the result in \( AL[i], Th \). In case of detecting a
source actor (an actor without any input channel except its self-loop channel) with an infinite throughput, the algorithm returns “no”, because this implies that its output channels are unbounded. The loop continues by setting $maxPTh$ to zero. This variable is a temporary variable for storing the maximum throughput of the predecessors of actor $AL[i]$ in iteration $i$. In the loop of lines 12 to 14, the minimum between the local throughput of actor $AL[i]$ and the minimum throughput of its predecessors is assigned to $AL[i].Th$. This value, according to Lemma 27, is the throughput of the actor $AL[i]$. Note that since the actors are topologically sorted in $AL$, the throughput of all predecessors has already been calculated. The maximum throughput of the predecessors of actor $AL[i]$ is assigned to $maxPTh$.

The test of line 15 checks whether the maximum throughput of predecessors of actor $AL[i]$ (excluding $AL[i]$) is greater than the throughput of actor $AL[i]$ itself. In case it is, according to Lemma 29, at least one channel connecting a predecessor of actor $AL[i]$ to $AL[i]$ is unbounded.

If the algorithm reaches line 17, then no unbounded channel has been detected, and the graph is live and self-timed bounded. According to Corollary 37 and the fact that the reduced SDFG is an HSDFG with all repetition-vector entries one, the value of $AL[i].Th$ for all actors $AL[i] \in A_H$ is equal to the throughput of $G$. The algorithm returns $AL[1].Th$. The emphasis of algorithm isLive&SelftimedBounded is on verifying liveness and self-timed boundedness of an SDFG, so it returns as soon as it detects that the graph is not live or not self-timed bounded. It can be easily adapted to compute the throughput for SDFGs which are not self-timed bounded as well.

6 Related Work

There are interesting similarities between SDFGs and Petri nets. In particular, there is a straightforward translation from SDFGs to a subclass of Petri nets, called weighted Marked Graphs and vice versa, where actors are transitions, and channels are places. Marked Graphs, also called T-Graphs are known to be the subclass of Petri nets that is most amenable to rigorous analysis. Thus, it makes sense to compare the results obtained in this paper with the corresponding results in the literature concerning Petri nets. We studied liveness in combination with three different definitions of boundedness (Definitions 17, 18 and 19) for (timed) SDFGs.

We do not know of any related results for boundedness as defined by Definition 17. The only result we know for this type of boundedness is in [15] which only introduces it without providing necessary and sufficient conditions, as we do.

For strict boundedness in the sense of Definition 18, the problem has been studied from different viewpoints in the Petri-net literature (see for an overview [7, 14]). In particular, [19] gives necessary and sufficient conditions for strict boundedness of live weighted Marked Graphs (our Theorem 25). Strict boundedness is also the only kind of boundedness which has been investigated formally in the literature on SDFGs themselves; Karp and Miller in their seminal paper [10] introduced computation graphs, which are slightly more general than SDFGs. They proved necessary and sufficient conditions for liveness and strict boundedness in their model. Their results as well as those in [19] correspond to those presented in this paper.

Our third definition of boundedness, self-timed boundedness (see Definition 19) is defined on timed SDFGs. Therefore, we need to compare it with time-enabled Petri nets. Petri nets have been extended with quantitative time in different ways, by adding timing information to places, transitions and/or tokens (see [4] for a survey). The timed Petri net model that comes closest to timed SDFGs is the “time Petri net” model originally defined by [13]. This extension of Petri nets associates a duration (delay) and a deadline to transitions. We are not aware of any study of the self-timed boundedness problem for the subclass of time Marked Graphs. In [17], the liveness and strict boundedness problem for time Petri nets is studied but only some sufficient conditions are given. These conditions guarantee that once a time Petri net satisfies certain syntactic constraints, it is live and strictly bounded if the underlying untimed Petri net is live and strictly bounded. Unfortunately, the results of [17] cannot be applied in our setting since the syntactic constraints require the absence of either duration or deadline both of which are necessary for translation of timed SDFGs to time Petri nets. [9] proves a general undecidability result for strict boundedness of time Petri net of [13]. However, in [2], two sufficient conditions are given for strict boundedness of time Petri nets. We are not aware of any result about self-timed boundedness as defined in Definition 19. To the best of our knowledge, both the concept and the derived results are novel.

7 Conclusions

We have studied the liveness and boundedness of Synchronous Data Flow Graphs, which are also known as weighted Marked Graphs in the Petri-net literature. Liveness and boundedness is a prerequisite of any meaningful SDFG model of a streaming multi-media application. Two known notions of boundedness, namely boundedness and strict boundedness, have been studied rigorously, and in particular necessary and sufficient conditions for liveness in combination with these two types of boundedness have been given. For strict boundedness, these conditions were already known from the Petri-net literature. Furthermore, a new notion, self-timed boundedness, was introduced. Self-
timed boundedness checks whether self-timed execution of an SDFG is bounded. A self-timed execution yields the maximum throughput for an SDFG. Necessary and sufficient conditions for self-timed boundedness and liveness have been proven. An algorithm for checking these conditions was presented. Besides, existing throughput analysis techniques, which are only valid for strongly connected graphs, are extended to arbitrary consistent SDFGs.

References


