

Formal Verification of Globally-Iterated/Locally-Non-Iterated Systems

Lubomir Ivanov

Department of Computer Science
Stevens Institute of Technology
Hoboken, NJ 07030
livanov@cs.stevens-tech.edu

Ramakrishna Nunna

Department of Electrical and Computer Engineering
California State University Fresno, MS 94
Fresno, CA 93740
rnunna@csufresno.edu

Abstract

Formal Verification of hardware has significantly gained in popularity as an alternative to testing and simulation in hardware design. Recently we introduced a new methodology for verification of non-iterated systems. The technique is based on the inductively defined notion of a series-parallel poset. In this paper we extend the notion of series-parallel posets to allow the modeling of systems involving global iteration. For this class of systems we present a verification algorithm, and discuss its foundation.

Introduction

An iterated system is a system which employs feedback in its own operation or the operation of one or more of its components. Our goal is to provide an approach that allows us to verify the occurrence of sequences of events during the operation of the system.

Examples of iterated systems at the hardware level include the following:

- a) At the logic gate level, any sequential circuit (e.g. a flip-flop), which involves feedback
- b) At the system level, any system which performs multiple iterations
- c) Various protocols for serial and parallel data communication, involving the repeated sending and receiving of data.

Series Parallel Posets:

In a recent paper [INB'99] we introduced the notion of series-parallel posets for verification of non-iterated systems.

A partially ordered set (poset) is a set with a reflexive, antisymmetric, and transitive relation defined on the set elements.

A Σ^* -labeled poset $P=(P, \leq, l)$ consists of a poset (P, \leq) , and an assignment of a nonempty word (i.e. a label) $l(v) \in \Sigma^*$ to each vertex v in P . We define two operations on labeled posets:

Given posets P and Q with $P \cap Q = \emptyset$,

Concatenation: $P \bullet Q := (P \cup Q, \leq_{P \bullet Q})$

Shuffle: $P \otimes Q := (P \cup Q, \leq_{P \otimes Q})$,

$$\begin{aligned} \text{where: } v \leq_{P \bullet Q} v' &\iff v \leq_P v' \vee v \leq_Q v' \vee \\ &\quad (v \in P \wedge v' \in Q) \\ v \leq_{P \otimes Q} v' &\iff v \leq_P v' \vee v \leq_Q v' \end{aligned}$$

A Series-Parallel Poset (SPP) is defined inductively as:

- The empty poset, I , is a SPP
- The singleton posets labeled σ , for each $\sigma \in \Sigma$
- If P and Q are SPPs, so are $P \bullet Q$ and $P \otimes Q$

The set of all series parallel posets, denoted $SP(\Sigma^*)$, closed under concatenation and shuffle, forms a bimonoid, with identity the empty poset I . Bloom and Esik proved that $SP(\Sigma^*)$ is freely generated in the variety of all bimonoids by the set Σ^* [BE'96].

In [INB'99] we showed how to define the behavior and properties of non-iterated systems through series-parallel posets. We also presented two verification predicates which test whether a property is satisfied within the behavior of a system.

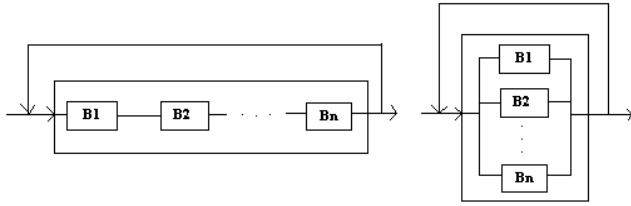
In the case of iterated systems, the behavior, B , of the system whose operation we would like to verify as well as the property, P , to be verified are defined as elements of the star shuffle semiring $\mathcal{S} = (\mathcal{S}, +, \bullet, \otimes, *, \emptyset, I)$ of series-parallel posets, defined, in turn, as follows:

- \mathcal{S} - the set of all finite subsets of $SP(\Sigma^*)$, closed under the semiring operations
- If $K \in \mathcal{S} \wedge L \in \mathcal{S}$, then $K + L = \{P \mid P \in K \vee P \in L\} \in \mathcal{S}$
- If $K \in \mathcal{S} \wedge L \in \mathcal{S}$, then $K \bullet L = \{P \bullet Q \mid P \in K \wedge Q \in L\} \in \mathcal{S}$
- If $K \in \mathcal{S} \wedge L \in \mathcal{S}$, then $K \otimes L = \{P \otimes Q \mid P \in K \wedge Q \in L\} \in \mathcal{S}$
- If $K \in \mathcal{S}$, then $K^* = I + K + K^2 + \dots = \sum_{i=0.. \infty} K^i \in \mathcal{S}$
- \emptyset is the empty set of posets
- I is the empty poset

Thus, the behavior $B \in \mathcal{S}$ and the verification property $P \in \mathcal{S}$, i.e. both the behavior and the property of an iterated system are sets of series-parallel posets.

Globally-Iterated/Locally-Non-Iterated Systems

Globally-Iterated/Locally-Non-Iterated Systems involve a number of components (sub-systems), which work independently or in series, but do not involve iteration themselves. Instead, when the final system output is computed, it is fed back for another iteration.



Globally-Iterated/Locally-Non-Iterated Systems with Components in Series and in Parallel

The behavior can be expressed in one of two forms:

- $B = (B_1 \bullet B_2 \bullet \dots \bullet B_n)^*$
- $B = (B_1 \otimes B_2 \otimes \dots \otimes B_n)^*$

where $B^* = I + B + B^2 + B^3 + \dots = \cup_{i=0, \dots, \infty} B^i$, and B is a series-parallel poset.

There are 4 property forms:

- $P = P_1 \bullet P_2 \bullet \dots \bullet P_n$
- $P = P_1 \otimes P_2 \otimes \dots \otimes P_n$
- $P = P_1 + P_2 + \dots + P_n$
- $P = P_1^*$

Analogously to the verification of properties of non-iterated systems, given two sets of series-parallel posets - a system behavior B , and a property P to verified - we define two verification predicates that check the proper embedding of the property in the behavior.

- a) **AS_GILNIS**(B, P) (which we shall read as “Property P is always satisfied within the behavior B of a globally-iterated/locally-non-iterated system”) is a binary predicate which takes two sets of series-parallel posets and verifies that the events in indicated in the property P always occur in the specified order within the system behavior, B .
- b) **SS_GILNIS**(B, P) (which we shall read as “Property P is sometimes satisfied within the behavior B of a globally-iterated/locally-non-iterated system”) is a binary predicate which takes two sets of series-

parallel posets and verifies that the events in property P can sometimes occur in the specified order within the system behavior, B .

The verification questions will be formulated as well-formed formulas (wffs) over series-parallel posets. The atomic formulas are given by the two predicates, and the non-atomic wffs are formed from the atomic ones using the logic connectives AND (\wedge), OR (\vee), and NOT (\neg).

Reduction of the Behavior

In most practical cases, the behavior of a real system will involve thousands and sometimes even millions of distinct events. A typical property, on the other hand, will only contain on the order of tens of events or less. Since the performance of our algorithm depends on the size of the behavior and the property expressions, any improvement to the performance of the verification procedure has to involve a reduction of the size of the behavior.

On the other hand, the presence or absence of other system events not specified in the property does not influence the outcome of the verification procedure. That leads to the natural idea of "removing" or "ignoring" the events specified in the behavior which do not appear in the verification property. To accomplish this, we need to introduce an appropriate function which will reduce the overall system behavior to one containing only the events specified in the verification property. This reduction significantly reduces the complexity of the verification algorithms, and significantly simplifies their analysis.

Let us introduce the projection function $Pr_GILNIS(B, set(P))$, which takes a behavior, B , and a set of events, and returns a reduced behavior, B' , with respect to the events in the specified set. The projection function is a mapping:

$$Pr_GILNIS : \mathcal{S} \times \Sigma^* \rightarrow \mathcal{S},$$

where \mathcal{S} is the set of all behaviors over an alphabet Σ , and Σ^* is the set of all subsets of events over Σ .

Pr_GILNIS is defined as follows:

$$Pr_GILNIS(A, \emptyset) = I$$

$$Pr_GILNIS(e_i, E) = \begin{cases} e_i & \text{if } e_i \in E \\ I & \text{otherwise} \end{cases}$$

$$Pr_GILNIS(A \bullet B, E) = Pr_GILNIS(A, E) \bullet Pr_GILNIS(B, E)$$

$$Pr_GILNIS(A \otimes B, E) = Pr_GILNIS(A, E) \otimes Pr_GILNIS(B, E)$$

$$Pr_GILNIS(A + B, E) = Pr_GILNIS(A, E) + Pr_GILNIS(B, E)$$

$$Pr_GILNIS(A^*, E) = (Pr_GILNIS(A, E))^*$$

where $A, B \in \mathcal{S}$, and $E \in \Sigma^*$

The effect of the projection function defined above is to substitute I in place of all events *not in the set* E .

Facts about the Behavior of Globally-Iterated/Locally-Non-Iterated Systems

First, observe that B^* implies that the system performs 1 or more, but an unspecified number of iterations, i.e. it's possible that it performs a single iteration, or two iterations, or three, etc. This nondeterminism regarding the precise number of iterations seems unnatural at first, but is perfectly suited for describing a variety of real-world systems. In the case of communication protocols, for example, the sender must perform a certain, unspecified number of iterations while waiting for the receiver to retrieve the data in the shared buffer. The receiver must also wait for an unspecified amount of time until new data arrives in the buffer from the sender.

Next, note that, after the first iteration, all events in the behavior B^* have already occurred, and, therefore, the occurrence of any event, e , in any subsequent iteration has all events in the behavior as predecessors, i.e. $\text{pred}_{B^*}(\{e\}) = B$ for all iterations $n > 1$.

Notice also that, with regard to the succession or independence of events, things are no different than in the case of non-iterated systems:

- Two sets of events are independent if no event in one of the sets triggers a chain of events leading to the occurrence of an event in the other set and vice versa, i.e. P_1 and P_2 are independent $\forall e \in P_2, \text{pred}_B(\{e\}) \cap \text{set}(P_1) = \emptyset \wedge \forall e \in P_1, \text{pred}_B(\{e\}) \cap \text{set}(P_2) = \emptyset$.
- P_1 (completely) precedes P_2 if all events in P_1 occur before any event in P_2 has occurred, i.e. $\forall e \in P_2, \text{pred}_B(\{e\}) \cap \text{set}(P_1) = \text{set}(P_1)$.
- P_1 partially precedes P_2 if the occurrence of any event in P_2 is triggered by the occurrence of at least one event in P_1 , i.e. $\forall e \in P_2, \text{pred}_B(\{e\}) \cap \text{set}(P_1) \neq \emptyset$.

The formal definitions of the verification predicates should not violate the above definitions of independence, concatenation, and partial concatenation.

One other issue we have to consider is the semantics of the verification predicates. The added complexity arising from iteration leads to multiple possible semantics, and thus we have to define our predicates very carefully in order to avoid ambiguities. We shall adopt the following interpretation of the verification predicates:

- **AS_GILNIS**(B, e): “Does the event e always occur once during a run of the system with behavior B ?”

- **SS_GILNIS**(B, e): “Does the event e sometimes occur once during a run of the system with behavior B ?”
- **AS_GILNIS**(B, e^*): “Is the event e always repeated (without any events occurring between two occurrences of e) during a run of the system with behavior B ?”
- **SS_GILNIS**(B, e^*): “Is the event e sometimes repeated (without any events occurring between two occurrences of e) during a run of the system with behavior B ?”

These single-event semantics naturally extend to (sets of) series-parallel posets.

As an example consider the following:

Example:

Let $B = (a \otimes b)^*$ and $P = a^* \bullet b^*$.

Is **SS_GILNIS**(B, P) TRUE?

The adopted semantics dictate the following interpretation of the property: “Event a is repeated, but no other events occur between two successive occurrences of a , and then b is repeated several times with no intervening occurrence of other events.” This “literal” interpretation leads to a FALSE answer

Based on the facts presented above we can make a few less trivial observations:

Theorem 1

Let $B = (B_1 \bullet B_2 \bullet \dots \bullet B_n)^*$ or $B = (B_1 \otimes B_2 \otimes \dots \otimes B_n)^*$ and $P = P_1^*$. Then **SS_GILNIS**(B, P) iff **SS_GILNIS**(B, P_1).

Theorem 2

If $P = P_1 \bullet P_2 \bullet \dots \bullet P_m$ or $P = P_1 \otimes P_2 \otimes \dots \otimes P_m$ and $\exists i \in [m]$ $P_i = P_{i1}^*$, then **SS_GILNIS**(B, P) is FALSE.

Corollary 1

If $P = P_1 \bullet P_2 \bullet \dots \bullet P_m$ or $P = P_1 \otimes P_2 \otimes \dots \otimes P_m$ and $\exists i \in [m]$ $P_i = P_{i1}^*$, then **AS_GILNIS**(B, P) is FALSE.

Theorem 3

Let $P = P_1 \otimes P_2 \otimes \dots \otimes P_m$ and $B = (B_1 \bullet B_2 \bullet \dots \bullet B_n)^*$. Then **SS_GILNIS**(B, P) is FALSE.

Corollary 2

Let $P = P_1 \otimes P_2 \otimes \dots \otimes P_m$ and $B = (B_1 \bullet B_2 \bullet \dots \bullet B_n)^*$. Then **AS_GILNIS**(B, P) is FALSE.

Theorem 4

Let $P = P_1 \otimes P_2 \otimes \dots \otimes P_m$ and $B = (B_1 \otimes B_2 \otimes \dots \otimes B_n)^*$. Then $SS_GILNIS(B, P)$ iff $SS_NI(B_1 \otimes B_2 \otimes \dots \otimes B_n, P)$ and $P \in SP(\Sigma^*)$.

Theorem 5

Let $P = P_1 \bullet P_2 \bullet \dots \bullet P_m$ and $B = B_{op}^*$ (where $B_{op} = (B_\bullet$ or $B_\otimes)$ and $B_\bullet = B_1 \bullet B_2 \bullet \dots \bullet B_n$ and $B_\otimes = B_1 \otimes B_2 \otimes \dots \otimes B_n$). Then $SS_GILNIS(B, P)$ iff $SS_NI(B_{op}, P)$ and $P \in SP(\Sigma^*)$.

Lemma 1

If $P \in SP(\Sigma^*)$ (i.e. P is "*" -free"), then $AS_GILNIS(B, P)$ is FALSE.

Theorem 6

Let $P = P_1^*$ and $B = B_{op}^*$ (where $B_{op} = (B_\bullet$ or $B_\otimes)$ and $B_\bullet = B_1 \bullet B_2 \bullet \dots \bullet B_n$ and $B_\otimes = B_1 \otimes B_2 \otimes \dots \otimes B_n$). Then $AS_GILNIS(B, P)$ iff $AS_NI(B_{op}, P_1)$.

Theorem 7

Let $P = P_1 \bullet P_2 \bullet \dots \bullet P_m$ or $P = P_1 \otimes P_2 \otimes \dots \otimes P_m$. Then $AS_GILNIS(B, P)$ is FALSE.

The Verification Predicates

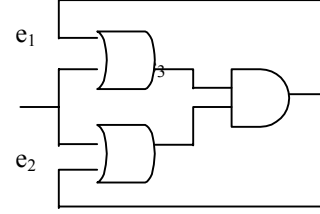
Now we are ready to give a formal definition of the verification predicates:

- a) $SS_GILNIS(B^*, P)$ iff :
- $|P| = 1 \wedge l(P) \in \Sigma_B$
 - $P = P_1^* \wedge (B = (B_1 \bullet B_2 \bullet \dots \bullet B_n)^* \vee B = (B_1 \otimes B_2 \otimes \dots \otimes B_n)^*) \wedge SS_GILNIS(B, P_1)$
 - $P = P_1 \otimes P_2 \otimes \dots \otimes P_m \wedge B = (B_1 \otimes B_2 \otimes \dots \otimes B_n)^* \wedge P \in SP(\Sigma^*) \wedge SS_NI(B_1 \otimes B_2 \otimes \dots \otimes B_n, P)$
 - $P = P_1 \bullet P_2 \bullet \dots \bullet P_n \wedge B = (B_1 \bullet B_2 \bullet \dots \bullet B_n)^* \wedge P \in SP(\Sigma^*) \wedge SS_NI(B_1 \bullet B_2 \bullet \dots \bullet B_n, P)$
 - $P = P_1 \otimes P_2 \bullet \dots \bullet P_n \wedge B = (B_1 \otimes B_2 \otimes \dots \otimes B_n)^* \wedge P \in SP(\Sigma^*) \wedge SS_NI(B_1 \otimes B_2 \otimes \dots \otimes B_n, P)$
 - $P = P_1 + \dots + P_n \wedge \exists i > 0, SS_GILNIS(B, P_i)$
- b) $AS_GILNIS(B^*, P)$ iff :
- $|P| = 1 \wedge l(P) \in \Sigma_B$
 - $P = P_1^* \wedge B = (B_1 \bullet B_2 \bullet \dots \bullet B_n)^* \wedge AS_NI(B_1 \bullet B_2 \bullet \dots \bullet B_n, P_1)$
 - $P = P_1^* \wedge B = (B_1 \otimes B_2 \otimes \dots \otimes B_n)^* \wedge AS_NI(B_1 \otimes B_2 \otimes \dots \otimes B_n, P_1)$
 - $P = P_1 + \dots + P_n \wedge \forall i > 0, AS_GILNIS(B, P_i)$

The worst case running time of the algorithms is bounded by $O(n)+O(m^3)$ and that of the average case - by $O(n)+O(m^2)$.

Modeling of GILNIS

Consider the following simple circuit:



The behavior of this circuit can be specified as:

$$B = ((e_1 \otimes e_2) \bullet e_3)^*$$

We can then verify properties such as:

a) "e₁ sometimes occurs before e₂", i.e.

$$SS_GILNIS(B, e_1 \bullet e_2)$$

b) "e₃ always occurs after e₂", i.e.

$$AS_GILNIS(B, e_2 \bullet e_3)$$

Verification Framework

Testing for the occurrence or non-occurrence of an event during system operation is a complex time consuming task. In current CAD methodologies, test vectors are typically used during simulation runs to look for event occurrences. However for each event, one or more test vectors are necessary - implying multiple simulations. Test Vector generation is furthermore a non-trivial task. Our approach on the other hand is able to check for the occurrence of multiple events during a single simulation run. Instead of artificial test vectors, real benchmarks or user programs can be run on the unit under test - and in conjunction with the modeling of the simulation data as series-parallel posets, we can easily verify the occurrence or non-occurrence of a set of events within the given behavior. Hardware description languages such as VHDL or Verilog can be used as the simulation language with a back end engine mapping the simulation data onto a model based upon series parallel posets. Using the formalism presented in this paper, one can then check for any given property within the behavior.

References

- [INB'99] L.Ivanov, R.Nunna, S.Bloom, "Modeling and Analysis of Non-Iterated Systems: An Approach based upon Series-Parallel Posets", Proceedings of ISCAS'99, Orlando, FL, May 30th - June 2nd 1999
- [BE'96] S.Bloom, Z.Esik, "Free Shuffle Algebras in Language Varieties", Theoretical Computer Science 163 (1996), 55-89, Elsevier