10th International Conference on Application of Concurrency to System Design

Order-Independence of Vector-Based Transition Systems

Matthias Raffelsieper*, MohammadReza Mousavi*, and Hans Zantema*†

*Department of Computer Science
TU Eindhoven, P.O. Box 513
5600 MB Eindhoven, The Netherlands
Email: {M.Raffelsieper,M.R.Mousavi,H.Zantema}@tue.nl

†Institute for Computing and Information Sciences
Radboud University Nijmegen, P.O. Box 9010
6500 GL Nijmegen, The Netherlands

***Abstract*—Semantics of many specification languages, particularly those used in the domain of hardware, is described in terms of vector-based transition systems. In such a transition system, each macro-step transition is labeled by a vector of inputs. When performing a macro-step, several inputs may potentially change. Each macro-step can thus be decomposed in a number of micro-steps, taking one input change at a time into account. This is akin to an interleaving semantics, where a concurrent step is represented by an interleaving of its constituting components. We present abstract criteria on vector-based transition systems, which guarantee that the next state computation is independent of the order in which these micro-steps are executed. If our abstract criteria are satisfied by the semantic definition of a certain specification, then its state-space generation or exploration algorithm needs to only consider one representative among all possible permutations of such micro-steps. We demonstrate the applicability of our abstract criteria to the specification of transistor netlists.**

I. Introduction

Reactive systems specification usually concern the interaction patterns for arbitrary input events, of which the order of arrival is unknown: input events can occur in all possible orders and even simultaneously. This principle also applies to the specification of hardware systems, e.g., in specifications given in languages such as Verilog, SystemC and VHDL. The operational semantics of reactive systems in general, and the above-mentioned languages in particular, is usually specified in terms of Labeled Transition Systems (LTSs). Popular types of LTSs used in the hardware domain include Mealy- [1] and Moore machines [2]. In this paper, we focus on the latter type, i.e., Moore machines or vector-based transition systems, for presentation purposes. Vector-based transition systems are transition systems in which transitions represent a change in one or more inputs and states represent collectively the current state and the output of the system.

The operational semantics of the above-mentioned languages usually deals with simultaneous events in an interleaving fashion, i.e., simultaneous input changes are processed one by one. When analyzing the operational behavior of specifications, this interleaving pattern leads to the well-known combinatorial explosion of the state space. It is thus customary to investigate only one (some) order of arrival for input events, which is called confluence- or partial-order reduction in the literature [3]. To ensure that the neglected orders do not lead to any new state, called *order-independence* in this paper, some sufficient conditions are established and checked. In this paper, we study the problem of proving order-independence for vector-based transition systems and seek sufficient *and necessary* conditions for order-independence.

Related work: In [4], we presented a method to check whether the non-determinism caused by choosing an arbitrary order for evaluating inputs of User Defined Primitives (UDPs) in the language Verilog [5], for which we defined a formal semantics and already observed this possible non-determinism in [6], does not affect the output value. Note that the computation of a UDP's output value is deterministic as soon as the order of processing inputs is known. This paper generalizes the result of [4] to arbitrary (thus, possibly non-deterministic) vector-based transition systems and improves the order-independence criteria presented there.

The technique presented in this paper can be seen as computing the independence relation that is sufficient to perform partial order reduction [3]; but it goes beyond ordinary independence relations by proving that, for this particular type of transition systems, our criteria are also necessary for partial order reduction, i.e., violating them results in order dependent behavior leading to different states.

In [7], an application of dynamic partial-order reduction techniques is used to efficiently explore all possible execution runs of a test-suite (non-exhaustive analysis) for parallel SystemC processes. To this end, the code of parallel SystemC processes is analyzed and non-commutative transitions are detected. Subsequently, all possible permutations of non-commutative actions are considered. In [8], the approach

1550-4808/10 $26.00 © 2010 IEEE
DOI 10.1109/ACSD.2010.24

of [7] is enhanced with slicing techniques and combined with static partial order reduction techniques. Neither of the approaches reported in [7], [8] claim the minimality of the generated set of orders (or in our terms, the necessity of the order-independence criteria). Our approach, however, guarantees that for each order dependency, it is possible to reach different states from some initial state and thus, there is a formal justification for including both orders.

Our analysis shares some basic ideas with the analysis of confluence in the setting of term rewrite systems, see for example [9], [10] for an introduction. Generally speaking, a system is confluent if any two computations can be joined again after an arbitrary number of steps. This relation has already been observed in [11], where sufficient conditions for confluence of general transition systems are given. In contrast to our work, [11] requires the transition system to be deterministic, whereas we also allow non-determinism, i.e., multiple successor states that are labeled with the same input pattern. However, we require deadlock-freedom for every state and every input pattern, which is not required globally in [11]. Furthermore, one should note that our notion of order-independence is stronger than confluence; confluence only requires the existence of a state in which two computations can be re-joined, for order-independence however it is required that the state reached after any of the two computations is the same.

The rest of this paper is organized as follows: In Section II, we introduce some formal definitions concerning vector-based transition systems. In Section III, we present our main results. First, we show that in our setting a quadratic number of comparisons (in the number of inputs) is sufficient to prove or disprove order-independence, i.e., whether the order of evaluating changing inputs affects the computation. Second, we prove that different evaluations, which have different restrictions on triggering input changes, are equivalent when order-independence holds. Section IV then applies this method to a target application, namely transistor-level netlist descriptions. To illustrate the applicability of our method in practice, we present a case study for the transistor netlists of the Nangate Open Cell Library [12] with promising results. We conclude the paper in Section V.

II. Preliminaries

A *vector-based transition system* over a domain M of input values is represented by a 3-tuple $T = (S, I, \delta)$, where S is an arbitrary set of states, $I = M^m$ the set of input vectors of a given and fixed size m, and $\delta : S \times I \to \mathcal{P}(S)$ is the transition function of T, with $\mathcal{P}(S)$ denoting the power set of S. We do not consider any initial state, instead we assume that an evaluation might start in any arbitrary state. This is in line with the requirements of our application domain, as a hardware system boots up from an unknown state. Since we are only concerned with vector-based transition systems, we refer to them just as *transition systems*. A transition system is called *deterministic*, if for all $\vec{i} \in I$ and all $s \in S$, $|\delta(s, \vec{i})| = 1$. Otherwise, it is called *non-deterministic*. A transition system is called *deadlock-free* if for all $\vec{i} \in I$ and all $s \in S$ we have that $|\delta(s, \vec{i})| \geq 1$. We also write $s \xrightarrow{\vec{i}}_T s'$ iff $s' \in \delta(s, \vec{i})$, and we leave out the subscript T in case the transition system is clear from the context. The *composition* of two transitions $\xrightarrow{\vec{i}_1}$ and $\xrightarrow{\vec{i}_2}$, denoted by $s_0 \xrightarrow{\vec{i}_1} \circ \xrightarrow{\vec{i}_2} s_2$ for states $s_0, s_2 \in S$, is defined iff a state $s_1 \in S$ exists such that $s_0 \xrightarrow{\vec{i}_1} s_1 \xrightarrow{\vec{i}_2} s_2$.

For a value $v \in M$, we define a *substitution* $\sigma = [a := v]$, where $1 \leq a \leq m$. Given a vector $\vec{v} \in M^k$, such a substitution is defined to be the mapping $\sigma(\vec{v}) = (w_1, \ldots, w_k)$, where for $i \neq a$, we have $w_i = v_i$ and $w_a = v$. Instead of $\sigma(\vec{v})$ we also write $\vec{v}\sigma$. We leave out the opening and closing parentheses of a vector when it causes no confusion.

The set $\mathcal{L}_k$ of all *lists* of numbers over the set $K = \{j \in \mathbb{N} \mid 1 \leq j \leq k\}$ is defined as the smallest set such that $\mathsf{nil} \in \mathcal{L}_k$ and $j : \ell \in \mathcal{L}_k$ for all $1 \leq j \leq k$ and all $\ell \in \mathcal{L}_k$. The function $\mathrm{sort} : \mathcal{L}_k \to \mathcal{L}_k$ returns its argument list sorted, i.e., for $\mathrm{sort}(\ell) = j_1 : \ldots j_r : \mathsf{nil}$ we have that $j_a \leq j_b$ for all $1 \leq a < b \leq r$. The length of a list $\ell \in \mathcal{L}_k$ is denoted $|\ell|$ and defined as $|\mathsf{nil}| = 0$ and $|j : \ell| = 1 + |\ell|$. To denote whether an element is contained in a list, we allow to interpret a list $\ell = j_1 : \cdots : j_r : \mathsf{nil}$ as a set and write $j \in \ell$ iff $j \in \{j_1, \ldots, j_r\}$. Finally, we define the set of *permutations* Π_k over the natural numbers between 1 and k as $\Pi_k = \{\ell \in \mathcal{L}_k \mid j \in \ell \text{ for all } 1 \leq j \leq k \text{ and } |\ell| = k\}$.

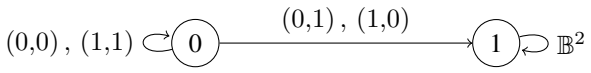
III. Order-Independence

As stated in the introduction, the order of arrival for input events, i.e., new values for inputs, is arbitrary and even several inputs may receive new values simultaneously. In the interleaving semantics, however, simultaneous events are considered one by one. Therefore, we interpret arbitrary transition systems in an interleaving fashion by studying their *one-input restricted* traces, defined below.

Definition 1. A (possibly infinite) trace $s_1 \xrightarrow{\vec{i}_1} s_2 \xrightarrow{\vec{i}_2} s_3 \xrightarrow{\vec{i}_3} \cdots$ is called *one-input restricted*, iff for all $j \in \mathbb{N} \setminus \{0\}$ we have that $\vec{i}_{j+1} = \vec{i}_j[k := v]$ for some $1 \leq k \leq m$ and some value $v \in M$.

The following example shows that, in general, all possible interleavings of input changes can differ from changing all inputs at the same time. This is because a one-input restricted trace still sees some old input values and gradually updates the state values.

Example 2. Let $T = (S, I, \delta)$ be the following deterministic transition system, where $S = \mathbb{B} = \{0, 1\}$, $I = \mathbb{B}^2$, and the transition function δ is illustrated below (the transition labeled $\mathbb{B}^2$ is a shorthand for the 4 transitions each with a label in $\mathbb{B}^2$):



We consider the different traces when transitioning from input vector $(0,0)$ to $(1,1)$:

$$\begin{array}{llllll} (0) & \xrightarrow{(0,0)} & (0) & \xrightarrow{(1,0)} & (1) & \xrightarrow{(1,1)} & (1) \\ (0) & \xrightarrow{(0,0)} & (0) & \xrightarrow{(0,1)} & (1) & \xrightarrow{(1,1)} & (1) \\ (0) & \xrightarrow{(0,0)} & (0) & \xrightarrow{(1,1)} & (0) & & \end{array}$$

Here, we see that both of the possible one-input restricted traces lead to the same final state (1), whereas the trace directly applying both of the new values results in a different final state (0).

We confine ourselves to the interleaving semantics in the remainder of this paper and hence, for each vector-based transition system T, we define its interleaving interpretation T^I, which, by construction, triggers at most one input in each step.

Definition 3. Let $T = (S, I, \delta)$ be a vector-based transition system with $I = M^m$. We define a corresponding one-input restricted vector-based transition system $T^I = (S \times I, \{1, \ldots, m\} \times M, \delta^I)$, where we also denote $(s, \vec{i}) \in S \times I$ by $s;\vec{i}$ and where the transition function $\delta^I : (S \times I) \times \{1, \ldots, m\} \times M \to \mathcal{P}(S \times I)$ is defined as $\delta^I(s;\vec{i}, j, v) = \{s';\vec{i}[j := v] \mid s' \in \delta(s, \vec{i}[j := v])\}$. We denote by $s;\vec{i} \xrightarrow{j} s';\vec{i'}$ that $s';\vec{i'} \in \delta^I(s;\vec{i}, j, v)$, where $v = i'_j$ for $\vec{i'} = (i'_1, \ldots, i'_m)$; note that by looking at the position j in $\vec{i}$ and $\vec{i'}$, we can recover the change in the input (second component v of the label). Hence, we only show the position on the label and leave out the actual value of the changing input.

This definition allows us to represent a one-input restricted trace in the form $s_1;\vec{i}_1 \xrightarrow{j_1} s_2;\vec{i}_2 \xrightarrow{j_2} \ldots$, where the numbers $j_1, j_2, \ldots$ indicate the triggered inputs in the corresponding steps. Note that using the construction given above, any one-input restricted trace of the transition system T can be translated into a trace of T^I and vice versa. Hence, we will use transition system T and its corresponding one-input restricted transition system T^I interchangeably in the following. In the following example, we present the transition system T^I for the transition system T of Example 2.

Example 4. For the transition system $T = (S, I, \delta)$ of Example 2, we have $T^I = (S \times I, I, \delta^I)$, where $S \times I = \mathbb{B} \times \mathbb{B}^2$ and δ^I is defined as shown in Figure 1 (dashed and bold lines are to be considered just like normal lines; we use them to illustrate other concepts in the subsequent examples). For the sake of brevity, we concatenate the input vector components. Furthermore, it should be mentioned that a label such as $1, 2$ does not denote a vector, but is a representation of the two transitions labeled by 1 and 2, respectively.

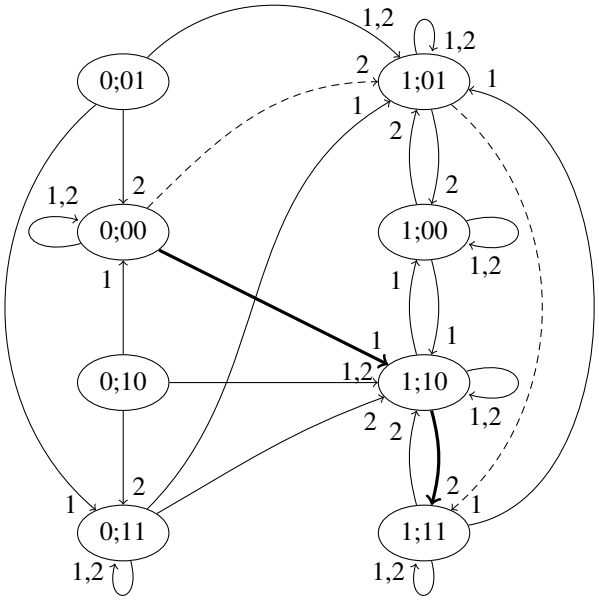


Figure 1. Transition System T^I for Example 2

To illustrate the construction that leads to the above graph, we consider the state $0;00$, which represents the state 0 in T in which we arrived with the input vector 00. When we now consider the successor states of this state when changing the first input, then we have two possibilities, either setting it to 0 or to 1. When setting it to 0, we have the same input vector 00, and since $0 \xrightarrow{00}_T 0$ we get the transition $0;00 \xrightarrow{1} 0;00$ in T^I. When setting the first input to 1, we obtain the input vector 10 which implies in T^I the transition $0;00 \xrightarrow{1} 1;10$, because in T we have the transition $0 \xrightarrow{10}_T 1$.

Next, we combine one-step transitions of the interleaving semantics to obtain different permutations of applying input. Hence, the transition relation introduced below is labeled with a permutation of positions to indicate the order in which the inputs are triggered.

Definition 5. Let $T = (S, I, \delta)$ be a transition system, $s, s' \in S$, $\vec{i^p}, \vec{i} \in I = M^m$, and $\ell \in \Pi_m$ with $\vec{i^p} = (i^p_1, \ldots, i^p_m)$, $\vec{i} = (i_1, \ldots, i_m)$, and $\ell = j_1 : \ldots : j_m : \mathsf{nil}$. For the transition system T^I, we define the transition relation $\twoheadrightarrow$ labeled by the permutation ℓ as $s;\vec{i^p} \overset{\ell}{\twoheadrightarrow} s';\vec{i}$ if and only if there exist states $s_0;\vec{i}_0, \ldots, s_{m+1};\vec{i}_{m+1} \in S \times I$ and a $1 \leq k \leq m$ such that $s_0;\vec{i}_0 \xrightarrow{k} s;\vec{i^p}$ and $s;\vec{i^p} = s_1;\vec{i}_1 \xrightarrow{j_1} s_2;\vec{i}_2 \xrightarrow{j_2} \ldots \xrightarrow{j_{m-1}} s_m;\vec{i}_m \xrightarrow{j_m} s_{m+1};\vec{i}_{m+1} = s';\vec{i}$.

This is not the only way to define the combination of considering input events; we could allow for applying a certain event zero or more times. We show in the remainder of this section that all different variants of the above definition

lead to the same conclusion as far as order-independence is concerned. Furthermore, it should be noted that the above definition requires the existence of a predecessor state $s_0;\vec{i}_0$ that reaches the state $s;\vec{i}^p$ with the old input vector $\vec{i}^p$. This requirement is added to rule out transient "boot-up" states that can only occur once, without any possibility of reaching them back again. Also, this requirement rules out input vectors that could not have been used to arrive at some state in a one-input restricted trace.

To illustrate the relation $\twoheadrightarrow$, we consider the state 0;00 in the transition system T^I of Example 4. We first observe that the state 0;00 is reachable, for example by $0;00 \xrightarrow{1} 0;00$. Hence, we have $0;00 \xrightarrow{1:2:\mathsf{nil}} 1;11$ due to the bold path in Example 4. Furthermore, we also have $0;00 \xrightarrow{2:1:\mathsf{nil}} 1;11$ because of the dashed path in Example 4. Hence, in this case it does not matter in which order we evaluate the changing inputs. In general, we want to determine this for all states and all possible orders of input changes, i.e., whether the state after applying a number of input changes is the same regardless of the chosen order of their application. If this is the case, we call the transition system *order-independent*.

Definition 6. Given a transition system $T = (S, I, \delta)$, relation $\twoheadrightarrow$ is called *order-independent*, iff $\xrightarrow{\ell} = \xrightarrow{\ell'}$ for all $\ell, \ell' \in \Pi_m$.

However, it can easily be seen that we do not have to consider all pairs of lists, since by transitivity of equality we can fix one list, e.g., to be the sorted list.

Lemma 7. *For transition system $T = (S, I, \delta)$, relation $\twoheadrightarrow$ is order-independent, iff $\xrightarrow{\ell} = \xrightarrow{\mathrm{sort}(\ell)}$ for all $\ell \in \Pi_m$.*

Proof: Follows from the transitivity of equality. ∎

Hence, one could check order-independence of $\twoheadrightarrow$ by constructing all of these $m!$ relations and comparing them for equality. However, following our basic idea in [4], we would like to reduce this check to a quadratic number of comparisons. The approach relies on the structure of the computation, only treating one input change at a time, and relying on two properties of the functions computing the next output value when considering a single input as changed: The first property is that when there is no change in the considered input, then also the output of that evaluation step remains unchanged. In other words, we require that the twofold application of the same input vector results in the same state as applying the input vector only once. Expressing this formally, we require for a transition system T the following property.

Fixed-Point Property. *Let $T = (S, I, \delta)$ be a transition system. T has the* fixed-point property *iff for all $s_1, s_2, s_3 \in S$, all $1 \le j, k \le m$, and all $\vec{i}^p, \vec{i} \in I$: If $s_1;\vec{i}^p \xrightarrow{j} s_2;\vec{i} \xrightarrow{k} s_3;\vec{i}$, then $s_3 = s_2$.*

The second (rather modest) property that we require in order to check order-independence efficiently is deadlock freedom. In our target application area of hardware descriptions, this is very natural since a hardware circuit will always compute some values from its inputs. For a deadlock-free transition system satisfying the fixed-point property, we have that order-independence is equivalent to checking the equality of two specific relations for all pairs of inputs. This is what we call the one-step reachable diamond property, which is defined next.

Definition 8. Let $T = (S, I, \delta)$ be a vector-based transition system with $I = M^m$.

Two inputs $1 \le j, k \le m$ with $j \ne k$ are said to have the *one-step reachable diamond property*, denoted $j \overset{\downarrow}{\diamond} k$, iff $s;\vec{i}^p \xrightarrow{j} \circ \xrightarrow{k} s';\vec{i} \iff s;\vec{i}^p \xrightarrow{k} \circ \xrightarrow{j} s';\vec{i}$ for all $s;\vec{i}^p, s';\vec{i} \in S \times I$ such that $s_0;\vec{i}_0 \xrightarrow{b} s;\vec{i}^p$ for some $s_0;\vec{i}_0 \in S \times I$ and $1 \le b \le m$.

This property is similar to the diamond property in [4], except that we also require one-step reachability of the state that starts the diamond. Using the one-step reachable diamond property we can now present our main theorem, showing that the one-step reachable diamond property is equivalent to order-independence of the relation $\twoheadrightarrow$ for a deadlock-free transition system. Hence, we can check the global property of order-independence by only considering a local property, namely the one-step reachable diamond property.

Theorem 9. *Let $T = (S, I, \delta)$ be a deadlock-free vector-based transition system having the fixed-point property, and let $I = M^m$. Then the transition relation $\twoheadrightarrow$ is order-independent, iff $j \overset{\downarrow}{\diamond} k$ for all $1 \le j < k \le m$.*

Proof: To show the "if"-direction, we assume that $j \overset{\downarrow}{\diamond} k$ holds for each pair $1 \le j < k \le m$ and prove that the property of Lemma 7 holds, i.e., that $\xrightarrow{\ell} = \xrightarrow{\mathrm{sort}(\ell)}$ for each $\ell \in \Pi_m$. To this end, let $\ell = j_1 : \cdots : j_m : \mathsf{nil}$. We proceed with an induction on the number of swaps required in the Bubble-Sort algorithm to sort list ℓ.

If no swaps are required, then $\ell = \mathrm{sort}(\ell)$ and therefore the theorem vacuously holds.

Otherwise, let $\ell' = j_1 : \cdots : j_{r+1} : j_r : \cdots : j_m : \mathsf{nil}$ be the list obtained after the first swap performed by the Bubble-Sort algorithm. Using the one-step reachable diamond property, we obtain for all one-step reachable states

$$\xrightarrow{j_r} \circ \xrightarrow{j_{r+1}} \;=\; \xrightarrow{j_{r+1}} \circ \xrightarrow{j_r}. \tag{1}$$

Let $s_1;\vec{i}^p \xrightarrow{\ell} s_2;\vec{i}$ for some $s_1;\vec{i}^p, s_2;\vec{i} \in S \times I$. This is by Definition 5 equivalent to the existence of a state $s_0;\vec{i}_0 \in S \times I$ and a $1 \le b \le m$ such that $s_0;\vec{i}_0 \xrightarrow{b} s_1;\vec{i}^p$ and $s_1;\vec{i}^p \xrightarrow{j_1} \circ \cdots \circ \xrightarrow{j_m} s_2;\vec{i}$. To this trace we apply (1), showing the

following equivalences:

$$\begin{aligned} & s_1;\vec{i}^p \xrightarrow{\ell} s_2;\vec{i} \\ \iff & s_1;\vec{i}^p \xrightarrow{j_1} \circ \cdots \circ \xrightarrow{j_r} \circ \xrightarrow{j_{r+1}} \circ \cdots \circ \xrightarrow{j_m} s_2;\vec{i} \\ \overset{(1)}{\iff} & s_1;\vec{i}^p \xrightarrow{j_1} \circ \cdots \circ \xrightarrow{j_{r+1}} \circ \xrightarrow{j_r} \circ \cdots \circ \xrightarrow{j_m} s_2;\vec{i} \\ \iff & s_1;\vec{i}^p \xrightarrow{\ell'} s_2;\vec{i} \end{aligned}$$

Therefore, we have $\xrightarrow{\ell} = \xrightarrow{\ell'}$. Applying the induction hypothesis to ℓ' gives us $\xrightarrow{\ell'} = \xrightarrow{\mathrm{sort}(\ell')}$, and since $\mathrm{sort}(\ell) = \mathrm{sort}(\ell')$ we have proven $\xrightarrow{\ell} = \xrightarrow{\mathrm{sort}(\ell)}$ as required.

To show the "only-if"-direction, assume towards a contradiction for some $1 \leq j < k \leq m$ we have $j \overset{\downarrow}{\not\diamond} k$, i.e., there exist some $s_0;\vec{i}_0$, $s;\vec{i}^p$, $s_1;\vec{i} \in S \times I$ and some $1 \leq b \leq m$ such that $s_0;\vec{i}_0 \xrightarrow{b} s;\vec{i}^p$, $s;\vec{i}^p \xrightarrow{j} \circ \xrightarrow{k} s_1;\vec{i}$, and not $s;\vec{i}^p \xrightarrow{k} \circ \xrightarrow{j} s_1;\vec{i}$ (or vice versa, but that case is symmetric to the considered one by exchanging the indices j and k).

Define lists $\ell = j : k : \ell_{\mathrm{tl}}$ and $\ell' = k : j : \ell_{\mathrm{tl}}$, where $\ell_{\mathrm{tl}} = 1 : \cdots : j-1 : j+1 : \cdots : k-1 : k+1 : \cdots : m : \mathsf{nil}$. Then, both ℓ and ℓ' are permutations by construction. Because the transition system is deadlock-free, there exists a state $s'_1;\vec{i} \in S \times I$ such that $s;\vec{i}^p \xrightarrow{\ell} s'_1;\vec{i}$, i.e., $s;\vec{i}^p \xrightarrow{j} \circ \xrightarrow{k} s_1;\vec{i} \xrightarrow{\ell_{\mathrm{tl}}} s'_1;\vec{i}$, where we abbreviate the relation $\xrightarrow{1} \circ \cdots \circ \xrightarrow{j-1} \circ \xrightarrow{j+1} \circ \cdots \circ \xrightarrow{k-1} \circ \xrightarrow{k+1} \circ \cdots \circ \xrightarrow{m}$ with $\xrightarrow{\ell_{\mathrm{tl}}}$. We can apply the fixed-point property repeatedly for the steps of $\xrightarrow{\ell_{\mathrm{tl}}}$ and therefore get that $s_1 = s'_1$. Assume $s;\vec{i}^p \xrightarrow{\ell'} s_1;\vec{i}$. Then there exist states $s'_0;\vec{i'}_0$, $s_2;\vec{i} \in S \times I$ and $1 \leq b' \leq m$ with $s'_0;\vec{i'}_0 \xrightarrow{b'} s;\vec{i}^p$ and $s;\vec{i}^p \xrightarrow{k} \circ \xrightarrow{j} s_2;\vec{i} \xrightarrow{\ell_{\mathrm{tl}}} s_1;\vec{i}$. Applying the fixed-point property repeatedly to this trace gives us $s_2 = s_1$. This however contradicts the assumption that $s;\vec{i}^p \xrightarrow{k} \circ \xrightarrow{j} s_1;\vec{i}$ does not hold, which was to be proven. ∎

Note that the proof, unlike the corresponding proof in [4], requires only deadlock-freedom and the fixed-point property, it does not need the property that a new input value, for an input that is not currently considered, does not change the computation. This is the case because we do not use completely new input vectors but only change the old vector at the positions of the currently considered pair in Definition 8 of the one-step reachable diamond property.

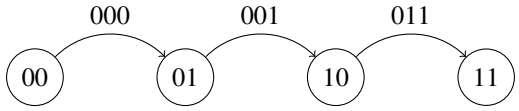
Both deadlock-freedom and the fixed-point property are only needed in the "only-if"-direction of the proof, i.e., the "if"-direction holds for arbitrary non-deterministic transition systems. This is expressed in the following corollary.

Corollary 10. *Given a transition system $T = (S, I, \delta)$ with $I = M^m$, the relation $\twoheadrightarrow$ is order-independent, if $j \overset{\downarrow}{\diamond} k$ for all $1 \leq j < k \leq m$.*

Proof: Follows from the proof of Theorem 9. ∎

Next, we give counter-examples witnessing that deadlock-freedom and the fixed-point property are required for the "only-if"-direction of Theorem 9. We start with a counter-example for dropping deadlock-freedom.

Example 11. Let $T = (\mathbb{B}^2, \mathbb{B}^3, \delta)$ be the transition system whose transition function δ is depicted below, where we concatenate the components of the state and input vectors.

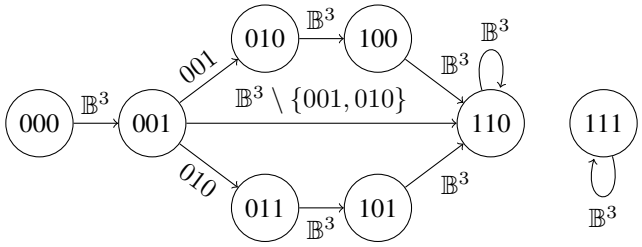


This transition system satisfies the fixed-point property, since in any state an input vector leading to that state cannot be applied again. Furthermore, it is order-independent, since there is no path of length 4 and hence any trace starting in a one-step reachable state will deadlock. However, $2 \overset{\downarrow}{\diamond} 3$ is not satisfied: For example, in state 01 together with the input vector 000 we have that $00;001 \xrightarrow{3} 01;000$ and $01;000 \xrightarrow{3} 10;001 \xrightarrow{2} 11;011$. However, no state $s;\vec{i} \in S \times I$ exists such that $01;000 \xrightarrow{2} s;\vec{i}$, hence the requirement of Definition 8 is not satisfied.

Note that for the above example, it is crucial to have $I = \mathbb{B}^3$ and not $I = \mathbb{B}^2$ by removing the first input component. This is because if $I = \mathbb{B}^2$ was used, Definition 5 would only require paths of length 3 (the initial step and two changes of the inputs), hence the above would be a counterexample to order-independence.

The next example shows that also the fixed-point property is needed for the "only-if"-direction of Theorem 9.

Example 12. Let $T = (\mathbb{B}^3, \mathbb{B}^3, \delta)$ be the transition system whose transition function δ is defined as illustrated below:



This transition system is deterministic and hence deadlock-free, however it does not satisfy the fixed-point property, since for example $010;001 \xrightarrow{1} 100;001 \xrightarrow{2} 110;001$ and $100 \neq 110$. Furthermore, relation $\twoheadrightarrow$ is order-independent:

- For any state $s \in \mathbb{B}^3 \setminus \{(000), (111)\}$, any input vectors $\vec{i}^p, \vec{i} \in \mathbb{B}^3$, and any permutation $\ell \in \Pi_3$ we have that $s;\vec{i}^p \xrightarrow{\ell} 110;\vec{i}$.
- For any state $s \in \mathbb{B}^3$, any input vectors $\vec{i}^p, \vec{i} \in \mathbb{B}^3$, and any permutation $\ell \in \Pi_3$ we have $000;\vec{i}^p \not\xrightarrow{\ell} s;\vec{i}$,

since no $s';\vec{i'} \in \mathbb{B}^6$ and no $1 \leq b \leq m$ exist such that $s';\vec{i'} \xrightarrow{b} 000;\vec{i}^p$.

- For any input vectors $\vec{i}^p, \vec{i} \in \mathbb{B}^3$ and any permutation $\ell \in \Pi_3$ we have $111;\vec{i}^p \xrightarrow{\ell} \!\!\!\rightarrow 111;\vec{i}$.

However, for the state $001 \in \mathbb{B}^3$ we see that $000;001 \xrightarrow{3} 001;000$ and $001;000 \xrightarrow{1} 010;001 \xrightarrow{2} 100;011$, whereas $001;000 \xrightarrow{2} 011;010 \xrightarrow{1} 101;011$, which shows $1 \not\downarrow 2$.

Definition 5 restricts the lists indicating the order of triggering inputs to permutations of the natural numbers from 1 to m. A natural generalization is therefore to also allow inputs to be triggered more than once. This can be generalized even further by only requiring those inputs to be triggered at least once, whose values in the initial and the final input vectors are different. Both of these generalizations are formally defined below.

Definition 13. Let $T = (S, I, \delta)$ be a transition system with $I = M^m$, $s;\vec{i}$, $s';\vec{i'} \in S \times I$, and $\ell = j_1 : \cdots : j_k : \mathsf{nil} \in \mathcal{L}_m$ with $j \in \ell$ for all $1 \leq j \leq m$.

We define relation $\twoheadrightarrow'$ for the transition system T^I as $s;\vec{i} \xrightarrow{\ell} \!\!\!\rightarrow' s';\vec{i'}$ iff a state $s_0;\vec{i}_0 \in S \times I$ and $1 \leq b \leq m$ exist such that $s_0;\vec{i}_0 \xrightarrow{b} s;\vec{i}$ and $s;\vec{i} \xrightarrow{j_1} \circ \cdots \circ \xrightarrow{j_k} s';\vec{i'}$.

Relation $\twoheadrightarrow'$ is called *order-independent*, iff $\xrightarrow{\ell} \!\!\!\rightarrow' = \xrightarrow{\ell'} \!\!\!\rightarrow'$ for all $\ell, \ell' \in \mathcal{L}_m$ with $j \in \ell$ for all $1 \leq j \leq m$ and ℓ' being a permutation of ℓ.

Definition 14. Let $T = (S, I, \delta)$ be a transition system with $I = M^m$, let $s;\vec{i}$, $s';\vec{i'} \in S \times I$ where $\vec{i} = (i_1, \ldots, i_m)$ and $\vec{i'} = (i'_1, \ldots, i'_m)$, and let $\ell = j_1 : \cdots : j_k : \mathsf{nil} \in \mathcal{L}_m$ with $\{1 \leq j \leq m \mid i_j \neq i'_j\} \subseteq \ell$.

Relation $\twoheadrightarrow''$ is defined as $s;\vec{i} \xrightarrow{\ell} \!\!\!\rightarrow'' s';\vec{i'}$ iff a state $s_0;\vec{i}_0 \in S \times I$ and $1 \leq b \leq m$ exist such that $s_0;\vec{i}_0 \xrightarrow{b} s;\vec{i}$ and $s;\vec{i} \xrightarrow{j_1} \circ \cdots \circ \xrightarrow{j_k} s';\vec{i'}$.

The relation $\twoheadrightarrow''$ is called *order-independent*, iff $\xrightarrow{\ell} \!\!\!\rightarrow'' = \xrightarrow{\ell'} \!\!\!\rightarrow''$ for all $\ell, \ell' \in \mathcal{L}_m$ with $\{1 \leq j \leq m \mid i_j \neq i'_j\} \subseteq \ell$ and ℓ' being a permutation of ℓ.

It can easily be seen from the above definitions that $\twoheadrightarrow \subseteq \twoheadrightarrow' \subseteq \twoheadrightarrow''$. To illustrate these two more general relations, we have for instance in Example 4 that $0;00 \xrightarrow{1:2:2:\mathsf{nil}} \!\!\!\rightarrow' 1;11$ and $0;00 \xrightarrow{1:2:2:\mathsf{nil}} \!\!\!\rightarrow'' 1;11$, whereas $0;00 \not\xrightarrow{1:2:2:\mathsf{nil}} \!\!\!\rightarrow 1;11$ since $|1 : 2 : 2 : \mathsf{nil}| = 3 \neq 2$. Furthermore, $0;00 \xrightarrow{1:\mathsf{nil}} \!\!\!\rightarrow'' 1;10$ but $0;00 \not\xrightarrow{1:\mathsf{nil}} \!\!\!\rightarrow' 1;10$, since $|1 : \mathsf{nil}| = 1 < 2$.

We also want to to check order-independence of these two generalized relations. Here, we again note that it suffices to consider only a single list and compare it to its corresponding sorted list, as was already observed for the relation $\twoheadrightarrow$ in Lemma 7.

Lemma 15. *Relation $\twoheadrightarrow'$ is order-independent, iff $\xrightarrow{\ell} \!\!\!\rightarrow' = \xrightarrow{\mathrm{sort}(\ell)} \!\!\!\rightarrow'$ for all $\ell \in \mathcal{L}_m$ with $j \in \ell$ for all $1 \leq j \leq m$.*

Relation $\twoheadrightarrow''$ is order-independent, iff $\xrightarrow{\ell} \!\!\!\rightarrow'' = \xrightarrow{\mathrm{sort}(\ell)} \!\!\!\rightarrow''$ for all $\ell \in \mathcal{L}_m$ with $\{1 \leq j \leq m \mid i_j \neq i'_j\} \subseteq \ell$.

Proof: Follows from transitivity of equality. ∎

Again, we want to use the one-step reachable diamond property given in Definition 8 to check whether these two generalized relations are order-independent or not. Since $\twoheadrightarrow \subseteq \twoheadrightarrow' \subseteq \twoheadrightarrow''$, we observe that the "only-if" direction in the proof of Theorem 9 holds directly. Furthermore, the "if" direction of that proof does not make use of the restriction to permutations, hence it also holds for the relations $\twoheadrightarrow'$ and $\twoheadrightarrow''$. This allows us to conclude that if one of the transition relations is order-independent, then all are, provided the transition system is deadlock-free and satisfies the the fixed-point property. This is formally expressed in the lemma below. There and in the following we also denote the relations $\twoheadrightarrow$, $\twoheadrightarrow'$, and $\twoheadrightarrow''$ with $\twoheadrightarrow^{(0)}$, $\twoheadrightarrow^{(1)}$, and $\twoheadrightarrow^{(2)}$, respectively, to be able to quantify over the three different relations.

Lemma 16. *For a deadlock-free transition system $T = (S, I, \delta)$ that satisfies the fixed-point property, relation $\twoheadrightarrow^{(a)}$ with $0 \leq a \leq 2$ is order-independent, iff a $0 \leq b \leq 2$ exists such that $\twoheadrightarrow^{(b)}$ is order-independent.*

Proof: The "only-if" direction holds trivially. For the "if" direction, assume $\twoheadrightarrow^{(b)}$ is order-independent. Then $j \downarrow\!\!\diamond\, k$ holds for all $1 \leq j < k \leq m$, otherwise we would have a counterexample to order-independence of $\twoheadrightarrow$ due to the "only-if" direction of Theorem 9. This however also results in a counterexample for $\twoheadrightarrow'$ and $\twoheadrightarrow''$, since $\twoheadrightarrow \subseteq \twoheadrightarrow' \subseteq \twoheadrightarrow''$. Hence, since the proof of the "if" direction of Theorem 9 does not make use of the requirements imposed onto list ℓ we have order-independence of $\twoheadrightarrow^{(a)}$. ∎

We presented in Example 12 that the fixed-point property is necessary for order-independence of $\twoheadrightarrow$. This example still applies to $\twoheadrightarrow'$. For relation $\twoheadrightarrow''$ however, this is not a valid counterexample, since we cannot assume that a trace has a certain (minimal) length, hence the traces showing that the one-step reachable diamond property is violated in Example 12 is also a counterexample to order-independence of $\twoheadrightarrow''$. Indeed, the following Lemma shows that order-independence of $\twoheadrightarrow''$ only requires deadlock-freedom and the one-step reachable diamond property, i.e., the fixed-point property is not required.

Lemma 17. *For a deadlock-free transition system $T = (S, I, \delta)$ with $I = M^m$ the relation $\twoheadrightarrow''$ is order-independent, iff $j \downarrow\!\!\diamond\, k$ holds for all $1 \leq j < k \leq m$.*

Proof: The "if" direction follows from Corollary 10. To show the "only-if" direction, assume $s;\vec{i}^p \xrightarrow{j:k:\mathsf{nil}} \!\!\!\rightarrow'' s_1;\vec{i}$ and not $s;\vec{i}^p \xrightarrow{k:j:\mathsf{nil}} \!\!\!\rightarrow'' s_1;\vec{i}$ for some $s, s_1 \in S$, $\vec{i}^p = (i^p_1, \ldots, i^p_m), \vec{i} = (i_1, \ldots, i_m) \in I$. By Definition 13 and Definition 3 we have $\vec{i} = \vec{i}^p[j := i_j, k := i_k]$ and therefore

$\{1 \leq j \leq m \mid i^p_j \neq i_j\} = \{j, k\}$. Hence, we have a counterexample to order-independence of $\twoheadrightarrow''$. ■

However, in case the transition system is deadlock-free and satisfies the fixed-point property, then the relations $\twoheadrightarrow^{(a)}$ with $0 \leq a \leq 2$ are all equivalent to the relation $\xrightarrow{1:\cdots:m:\mathsf{nil}}\!\!\!\!\twoheadrightarrow$ as we will show in the theorem below.

Theorem 18. *Let $T = (S, I, \delta)$ be a deadlock-free transition system with $I = M^m$ satisfying the fixed-point property.*

If $\twoheadrightarrow^{(b)}$ is order-independent for some $0 \leq b \leq 2$, then $\xrightarrow{\ell}\!\!\!\!\twoheadrightarrow^{(a)} = \xrightarrow{1:\cdots:m:\mathsf{nil}}\!\!\!\!\twoheadrightarrow$ for all $0 \leq a \leq 2$ and all lists $\ell \in \mathcal{L}_m$ satisfying the requirements of $\xrightarrow{\ell}\!\!\!\!\twoheadrightarrow^{(a)}$.

Proof: Let $\twoheadrightarrow^{(b)}$ be order-independent for some $0 \leq b \leq 2$. Due to Lemma 16, all relations $\twoheadrightarrow^{(a)}$ are order-independent, i.e., $\xrightarrow{\ell}\!\!\!\!\twoheadrightarrow^{(a)} = \xrightarrow{\mathrm{sort}(\ell)}\!\!\!\!\twoheadrightarrow^{(a)}$ for all lists $\ell \in \mathcal{L}_m$ that satisfy the requirements of $\twoheadrightarrow^{(a)}$. Hence, for $a = 0$ the theorem holds trivially.

For the remaining cases, let $\ell = j_1 : \cdots : j_{|\ell|} : \mathsf{nil}$ be an arbitrary list satisfying the requirements of $\twoheadrightarrow^{(a)}$. We define $\ell' = j_1 : \cdots : j_{|\ell|} : j_{|\ell|+1} : \cdots : j_{|\ell|+k} : \mathsf{nil}$ such that $j \in \ell'$ for all $1 \leq j \leq m$. By requirement on the list ℓ, we have $\{1 \leq j \leq m \mid i_j \neq i^p_j\} \subseteq \ell$, thus $i_{j_{|\ell|+r}} = i^p_{j_{|\ell|+r}}$ for all $1 \leq r \leq k$. Furthermore, for any $s;\vec{i} \in S \times I$ we have due to deadlock-freedom that a $s';\vec{i'} \in S \times I$ exists such that $s;\vec{i} \xrightarrow{j_{|\ell|+r}} s';\vec{i'}$ for all $1 \leq r \leq k$. Hence, because $i_{j_{|\ell|+r}} = i^p_{j_{|\ell|+r}}$ and the state s was reachable with input $\vec{i}$, we have that $\vec{i'} = \vec{i}$ and therefore can apply the fixed-point property, yielding $s' = s$. Applying this for all $1 \leq r \leq k$, we get $\xrightarrow{\ell'}\!\!\!\!\twoheadrightarrow^{(a)} = \xrightarrow{\ell}\!\!\!\!\twoheadrightarrow^{(a)}$.

Because $\twoheadrightarrow^{(a)}$ is order-independent, we have that $\xrightarrow{\ell'}\!\!\!\!\twoheadrightarrow^{(a)} = \xrightarrow{\mathrm{sort}(\ell')}\!\!\!\!\twoheadrightarrow^{(a)}$. Note that ℓ', and therefore also $\mathrm{sort}(\ell')$, might contain duplicates. However, for any computation sequence of the form $s;\vec{i} \xrightarrow{j} s';\vec{i'} \xrightarrow{j} s'';\vec{i''}$ that occurs as part of $\xrightarrow{\mathrm{sort}(\ell')}\!\!\!\!\twoheadrightarrow^{(a)}$, we have that $\vec{i'} = \vec{i''}$, thus we can again apply the fixed-point property which gives us that $s' = s''$. Thus, we can remove all duplicates from $\mathrm{sort}(\ell')$, which results in the list $1 : \cdots : m : \mathsf{nil}$. Hence, we have proven the theorem, as now $\xrightarrow{\ell}\!\!\!\!\twoheadrightarrow^{(a)} = \xrightarrow{\ell'}\!\!\!\!\twoheadrightarrow^{(a)} = \xrightarrow{\mathrm{sort}(\ell')}\!\!\!\!\twoheadrightarrow^{(a)} = \xrightarrow{1:\cdots:m:\mathsf{nil}}\!\!\!\!\twoheadrightarrow^{(a)} = \xrightarrow{1:\cdots:m:\mathsf{nil}}\!\!\!\!\twoheadrightarrow$. ■

To evaluate a deadlock-free transition system satisfying the fixed-point property with an order-independent transition relation $\twoheadrightarrow^{(a)}$ for $a \in \{0, 1, 2\}$ and possibly changing lists ℓ, it therefore suffices to only evaluate with the single relation $\xrightarrow{1:\cdots:m:\mathsf{nil}}\!\!\!\!\twoheadrightarrow$. This especially allows to reduce evaluations with lists of arbitrary length to evaluation with the fixed length m. If the relation is also allowed to depend on the input values, then it even suffices to only consider the changed inputs once, as the unchanged ones do not affect the final state.

Corollary 19. *Let $T = (S, I, \delta)$ be a transition system with $I = M^m$, $s, s' \in S$, $\vec{i^p} = (i^p_1, \ldots, i^p_m), \vec{i} = (i_1, \ldots, i_m) \in I$, and $0 \leq a \leq 2$. Define $\ell_c = j_1 : \cdots : j_k : \mathsf{nil}$, where $\{j_1, \ldots, j_k\} = \{1 \leq j \leq m \mid i^p_j \neq i_j\}$.*

Then for all lists $\ell \in \mathcal{L}_m$ satisfying the requirements of $\xrightarrow{\ell}\!\!\!\!\twoheadrightarrow^{(a)}$, $s;\vec{i^p} \xrightarrow{\ell}\!\!\!\!\twoheadrightarrow^{(a)} s';\vec{i}$, iff $s;\vec{i^p} \xrightarrow{\ell_c}\!\!\!\!\twoheadrightarrow'' s';\vec{i}$.

Proof: Follows from Theorem 18, since $\xrightarrow{\ell}\!\!\!\!\twoheadrightarrow^{(a)} = \xrightarrow{1:\cdots:m:\mathsf{nil}}\!\!\!\!\twoheadrightarrow = \xrightarrow{\ell_c}\!\!\!\!\twoheadrightarrow''$ for all lists $\ell \in \mathcal{L}_m$ that satisfy the requirements of $\xrightarrow{\ell}\!\!\!\!\twoheadrightarrow^{(a)}$. ■

IV. Application to Netlists

Cell Libraries are collections of logic cores used to construct larger chip designs and consist of combinational cells (e.g., `nand` and `xor`) and sequential cells (e.g., latches and flip-flops). These cells are commonly described both as *transistor netlists*, specifying the implementation that is finally used for production, and as functional description in a hardware definition language, e.g., Verilog. For the subset of Verilog that is usually found in cell libraries we have defined a formal semantics in [6] and described in [4] how to check order-independence. In the remainder of this section, we present how the generic theory developed in the present paper can be used to also check order-independence of transistor netlist descriptions.

To check order-independence of a transistor netlist, we use their representation in terms of a set of *fixed-point equations*, using the method of [13]. To give a formal description of fixed-point equations, let V_S and V_I be two disjoint sets of variables, whose values are in some domain M. Furthermore, let $n, m \in \mathbb{N}$, $\vec{s^v} = (s^v_1, \ldots, s^v_n) \in V^n_S$ with $s^v_j \neq s^v_k$ for all $1 \leq j < k \leq n$, and $\vec{i^v} = (i^v_1, \ldots, i^v_m) \in V^m_I$ with $i^v_j \neq i^v_k$ for all $1 \leq j < k \leq m$. Then a set $\mathcal{E} = \{s^v_1 \equiv f_1(\vec{i^v}, \vec{s^v}), \ldots, s^v_n \equiv f_n(\vec{i^v}, \vec{s^v})\}$, with functions $f_j : M^m \times M^n \to M$ for $1 \leq j \leq n$ is called a set of *fixed-point equations*, iff all of these functions satisfy the following *local fixed-point property*, requiring for all $1 \leq j \leq n$, all $\vec{i} \in M^m$, and all $\vec{s} \in M^n$ that

$$f_j(\vec{i}, \vec{s}) = f_j(\vec{i}, (f_1(\vec{i}, \vec{s}), \ldots, f_n(\vec{i}, \vec{s}))).$$

We interpret such a set of fixed-point equations as a transition system $T(\mathcal{E}) = (M^n, M^m, \delta)$, where $\vec{s} \xrightarrow{\vec{i}}_{T(\mathcal{E})} \vec{s'}$, with $\vec{s'} = (s'_1, \ldots, s'_n)$, iff $s'_j = f_j(\vec{i}, \vec{s})$ for all $1 \leq j \leq n$. Again we leave out the subscript $T(\mathcal{E})$ if the set of fixed-point equations is clear from the context. Note that $T(\mathcal{E})$ is deterministic, i.e., for every $\vec{s}$ and every $\vec{i}$ there exists exactly one $\vec{s'}$ such that $\vec{s} \xrightarrow{\vec{i}} \vec{s'}$.

As an example, the set of fixed-point equations extracted from the transistor netlist of a D flip-flop is presented below.

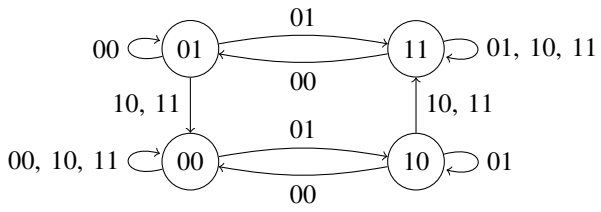


Figure 2. D flip-flop as a transition system

Example 20. We consider the following set of fixed-point equations modeling a D flip-flop, where $V_S = \{\mathrm{iq}, \mathrm{q}\}$ and $V_I = \{\mathrm{ck}, \mathrm{d}\}$.

$$\begin{aligned} \mathrm{iq} &\equiv \neg\mathrm{ck} \wedge \mathrm{d} \vee \mathrm{ck} \wedge \mathrm{iq} \\ \mathrm{q} &\equiv \mathrm{ck} \wedge \mathrm{iq} \vee \neg\mathrm{ck} \wedge \mathrm{q} \end{aligned}$$

These equations describe the transition system depicted in Figure 2, where the state variables are concatenated in the order iq, q and the inputs in the order ck, d.
We observe that this transition system is deterministic, hence deadlock-free, and satisfies the fixed-point property. Furthermore, it is order-dependent: for example, in state 00 we have that $00;10 \xrightarrow{1} 00;00$ and $00;00 \xrightarrow{2} 10;01 \xrightarrow{1} 11;11$, whereas $00;00 \xrightarrow{1} 00;10 \xrightarrow{2} 00;11$. This shows that it matters for a flip-flop whether first the data input d changes and then the clock ck, which corresponds to the first trace and sets the output q to the new value of input d, or vice versa, which corresponds to the second trace and sets the output q to the old value of input d.

For the special case of a transition system that stems from a set of fixed-point equations, the required global fixed-point property always holds as we will show next.

Lemma 21. *Every set of fixed-point equations has the fixed-point property.*

Proof: Let $\mathcal{E} = \{s_1^v \equiv f_1(\vec{i}^v, \vec{s}^v), \ldots, s_n^v \equiv f_n(\vec{i}^v, \vec{s}^v)\}$ be a set of fixed-point equations and let $\vec{s}_1;\vec{i}^p \xrightarrow{j} \vec{s}_2;\vec{i} \xrightarrow{k} \vec{s}_3;\vec{i}$. Assume that $\vec{s}_2 = (s_{2,1}, \ldots, s_{2,n}) \neq (s_{3,1}, \ldots, s_{3,n}) = \vec{s}_3$. Then $1 \leq j \leq n$ exists such that $s_{2,j} \neq s_{3,j}$.

By definition, we have that $\vec{s}_2 = (f_1(\vec{i}, \vec{s}_1), \ldots, f_n(\vec{i}, \vec{s}_1))$. Since $\mathcal{E}$ is a set of fixed-point equations, we furthermore have the following for the j-th component:

$$\begin{aligned} s_{2,j} &= f_j(\vec{i}, \vec{s}_1) \\ &= f_j(\vec{i}, (f_1(\vec{i}, \vec{s}_1), \ldots, f_n(\vec{i}, \vec{s}_1))) = f_j(\vec{i}, \vec{s}_2) \end{aligned}$$

Also by definition, we have $s_{3,j} = f_j(\vec{i}, \vec{s}_2)$ and hence $s_{3,j} = s_{2,j}$. This is a contradiction to our initial assumption, which proves the lemma. ∎

To apply the method to such a set of fixed-point equations, we encode the domain M, which is usually either the set of Booleans or the set of Booleans with an unknown value X, as Boolean vectors. Then we construct for each pair of inputs a pair of BDDs representing the two sides of the one-step reachable diamond property that are given in Definition 8. If all such pairs of BDDs are equal we have proved order-independence due to Theorem 9. Otherwise, we have found a set of counterexample states, which can be obtained by computing the XOR of the unequal BDDs. Particularly, for this application we found that including the one-step reachability into the requirement removes many spurious counterexamples, which were due to certain dependencies of the internal signals on the input signals. This corresponds to a stabilization of the netlist before applying the first input vector, i.e., all transistors are evaluated w.r.t. the previous input vector until there are no more changes.

However, usually some of the reported order-dependencies are expected and should not be deemed an error. Therefore, we allow the designer to rule out certain combinations of input patterns. For this purpose, we again use the format of Verilog timing checks, as in [4]. These timing checks describe behavior that is considered illegal, hence we remove counterexamples that contradict one of the timing checks specified for the currently investigated cell. This is implemented by constructing a BDD that describes all states violating one of the timing checks and then taking the conjunction of the counterexample states with the negation of this BDD. Thereby, we obtain another BDD that describes all counterexample states not violating any of the timing constraints, which are reported to the user. Note that this new BDD might describe an empty set of counterexample states; in this case, the only counterexamples that were found previously were in conflict with at least one timing check and therefore considered superfluous, i.e., the netlist is order-independent for all traces that respect the timing checks.

We have applied the method to the transistor netlists of the 12 sequential cells in the Nangate Open Cell Library [12], whose corresponding functional descriptions were used in [4]. For each of the cells we used the timing checks that were given in the corresponding Verilog module. With these timing checks ruling out illegal behavior, we were able to prove ten of these cells order independent, when considering the inputs to be binary. For two cells however, namely the cells `DFFRS` and `SDFFRS`, a counterexample was found. This counterexample is the same that was found for the Verilog implementation in [4]: there is no timing check specified for the deactivation of the set and reset inputs, hence when deactivating both at almost the same time the output value depends on whether the set is deactivated first leaving the reset still active, or whether the reset is deactivated first leaving the set still active. This problem might therefore really cause non-deterministic behavior, which is undesired. It can be solved by adding a timing check that disallows simultaneous disabling of both the set and reset input, then also our technique does not report any further order-dependencies.

For each of the 12 transistor netlists, checking order-independence took less than 0.25 seconds on a computer with an Intel Pentium 4 processor with 3.0 GHz and 1 GB memory. Therefore, it can for example be used as a fast preprocessing step to compute the independence relation needed for partial order reduction [3], which allows to reduce the state space that has to be explored when checking other properties.

V. Conclusion

In this paper, we presented a method to efficiently check order-independence of non-deterministic vector-based transition systems that are deadlock-free and that satisfy the fixed-point property, i.e., repeated application of the same input pattern does not change the current state. If a transition system is order-independent, then evaluations that change at most one input in each step are independent of the order of applying the input changes. Furthermore, if order-independence holds for a transition system, then an evaluation triggering multiple inputs for the same input vector a number of times is equivalent to the evaluation triggering only the changed inputs exactly once. We have applied our techniques to transistor-level descriptions of hardware cells and our experimental results show that this method can be used to identify problematic situations in the implementation. Finally, we would like to remark that the presented technique is also useful to check other (temporal) properties of a given transition system, as it computes the independence relation needed to perform partial order reduction [3].

We would like to further restrict our technique to consider only those states that represent the steady-state behavior of the hardware cell, i.e., to exclude the transient start-up phase of the circuit. A possibility is to restrict the analysis to the terminal strongly connected components (SCCs) of the transition system, i.e., SCCs that are minimal w.r.t. the reachability relation. Ideally, we would like to combine symbolic techniques for the computation of SCCs, e.g., the algorithm of [14], with our technique.

Acknowledgment

The original research question for this paper was posed by Fenix DA and in particular, by Chris Strolenberg, to whom we are most thankful. Also, we would like to thank the anonymous reviewers for making several fruitful remarks, especially suggesting the elegant proof of Theorem 18.

References

[1] G. H. Mealy, "A Method for Synthesizing Sequential Circuits," *Bell Systems Technical Journal*, vol. 34, pp. 1045–1079, 1955.

[2] E. F. Moore, "Gedanken-Experiments on Sequential Machines," *Annals of Mathematical Studies*, vol. 34, pp. 129–153, 1956.

[3] D. Peled, "Ten Years of Partial Order Reduction," in *Proceedings of the 10th International Conference on Computer Aided Verification (CAV 1998)*, ser. Lecture Notes in Computer Science, vol. 1427. Springer-Verlag, 1998, pp. 17–28.

[4] M. Raffelsieper, M. R. Mousavi, J.-W. Roorda, C. Strolenberg, and H. Zantema, "Formal Analysis of Non-Determinism in Verilog Cell Library Simulation Models," in *Proceedings of 14th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2009)*, ser. Lecture Notes in Computer Science, vol. 5825. Springer-Verlag, 2009, pp. 133–148.

[5] "IEEE Std 1364-2005: IEEE Standard for Verilog Hardware Description Language," IEEE Computer Society Press, 2006.

[6] M. Raffelsieper, J.-W. Roorda, and M. R. Mousavi, "Model Checking Verilog Descriptions of Cell Libraries," in *Proceedings of the Ninth International Conference on Application of Concurrency to System Design (ACSD 2009)*. IEEE Computer Society Press, 2009, pp. 128–137.

[7] C. Helmstetter, F. Maraninchi, L. Maillet-Contoz, and M. Moy, "Automatic Generation of Schedulings for Improving the Test Coverage of Systems-on-a-Chip," in *Proceedings of the 6th International Conference on Formal Methods in Computer Aided Design (FMCAD 2006)*. IEEE Computer Society Press, 2006, pp. 171–178.

[8] S. Kundu, M. Ganai, and R. Gupta, "Partial order reduction for scalable testing of SystemC TLM designs," in *Proceedings of the 45th annual Design Automation Conference (DAC 2008)*. ACM Press, 2008, pp. 936–941.

[9] F. Baader and T. Nipkow, *Term Rewriting and All That*. Cambridge University Press, 1998.

[10] Terese, *Term Rewriting Systems*. Cambridge University Press, 2003.

[11] R. M. Keller, "A fundamental theorem of asynchronous parallel computation," in *Proceedings of the Sagamore Computer Conference*, ser. Lecture Notes in Computer Science, vol. 24. Springer-Verlag, 1975, pp. 102–112.

[12] Nangate Inc., "Open Cell Library v2008_10 SP1," 2008, downloadable from http://www.nangate.com/openlibrary/.

[13] R. Bryant, "Boolean Analysis of MOS Circuits," *IEEE Transactions on Computer-aided Design*, vol. 6, no. 4, pp. 634–649, 1987.

[14] R. Bloem, H. N. Gabow, and F. Somenzi, "An Algorithm for Strongly Connected Component Analysis in $n \log n$ Symbolic Steps," in *Proceedings of the Third International Conference on Formal Methods in Computer-Aided Design (FMCAD 2000)*, ser. Lecture Notes in Computer Science, vol. 1954. Springer-Verlag, 2000, pp. 56–73.