Form Methods Syst Des (2012) 40:283–329
DOI 10.1007/s10703-012-0144-6

Constructive Boolean circuits and the exactness of timed ternary simulation

Michael Mendler · Thomas R. Shiple · Gérard Berry

Published online: 28 March 2012

Abstract We classify gate level circuits with cycles based on their stabilization behavior. We define a formal class of combinational circuits, the *constructive* circuits, for which signals settle to a unique value in bounded time, for any input, under a simple conservative delay model, called the *up-bounded non-inertial (UN) delay*. Since circuits with combinational cycles can exhibit asynchronous behavior, such as non-determinism or metastability, it is crucial to ground their analysis in a formal delay model, which previous work in this area did not do.

We prove that *ternary simulation*, such as the practical algorithm proposed by Malik, decides the class of *constructive* circuits. We prove that three-valued algebra is able to maintain correct and exact stabilization information under the UN-delay model, and thus provides an adequate electrical interpretation of Malik's algorithm, which has been missing in the literature. Previous work on combinational circuits used the *upbounded inertial (UI) delay* to justify ternary simulation. We show that the match is not exact and that stabilization under the UI-model, in general, cannot be decided by ternary simulation. We argue for the superiority of the UN-model for reasons of complexity, compositionality and electrical adequacy. The UN-model, in contrast to the UI-model, is consistent with the hypothesis that physical mechanisms cannot implement non-deterministic choice in bounded time.

As the corner-stone of our main results we introduce *UN-Logic*, an axiomatic specification language for UN-delay circuits that mediates between the real-time behavior and its abstract simulation in the ternary domain. We present a symbolic simulation calculus for

M. Mendler (✉)
Faculty of Information Systems and Applied Computer Sciences, The Otto-Friedrich University of Bamberg, Bamberg, Germany
e-mail: michael.mendler@uni-bamberg.de

T.R. Shiple
Synopsys, Inc., Mountain View, CA, USA
e-mail: Thomas.Shiple@synopsys.com

G. Berry
INRIA, Sophia Antipolis, France
e-mail: gerard.berry@sophia.inria.fr

circuit theories expressed in UN-logic and prove it sound and complete for the UN-model. This provides, for the first time, a correctness and exactness result for the timing analysis of cyclic circuits. Our algorithm is a timed extension of Malik's pure ternary algorithm and closely related to the timed algorithm proposed by Riedel and Bruck, which however was not formally linked with real-time execution models.

Keywords Combinational circuits · Delay models · Ternary simulation · Constructive logic

1 Introduction

We analyze the logical behavior of combinational circuits described at the gate level. Informally, we say that a circuit is *combinational* if for every input, the output stabilizes to a unique value within a bounded amount of time. All acyclic circuits are combinational in this sense and their analysis is straightforward [47]: a Boolean function for each node, in terms of the circuit inputs, can be derived by applying functional composition in a topological order and these Boolean functions exactly correspond to the steady-state electrical behavior of the circuit. On the other hand, circuits with feedback cycles are usually avoided because the presence of cycles can lead to unpredictable memory effects such as oscillation, metastability or nondeterminism. However, not all cycles lead to undesirable behavior, as some cyclic circuits are combinational. Such cycles are known as *false* cycles, i.e., for every input provided by the operating environment, no event can be propagated around the cycle [23, 41]. Techniques to analyze cyclic combinational circuits are useful because they arise in practical situations. Consider the following:

1. High-level synthesis, where cycles are created to share data-path resources. An example is Fig. 1, which computes $z = \mathit{if}\ (c)\ \mathit{then}\ F(G(x))\ \mathit{else}\ G(F(x))$. Even though the cycle in the example is false (because c and $\overline{c}$ are mutually exclusive), Stok [45] notes that such cycles are undesirable because downstream tools cannot handle cyclic circuits. He solves this problem by modifying resource sharing to prevent cycles from being created, at the cost of more control circuitry. Our philosophy is to provide rigorous analysis so that cyclic circuits can be handled directly.
2. The composition of Mealy machines. When a single FSM is synthesized within the context of a set of interacting FSMs, the resulting composition may create combinational cycles [31, 49].
3. The specification of reactive programs in synchronous programming languages. A language like Esterel [4] allows the specification of "zero-delay cycles", and it is the task of the compiler to determine if such cycles are false.
4. The design of symmetric protocols or resource access strategies, where cycles arise naturally [12]. However, the cycles are false for all reachable states of the design. The analysis of such circuits is discussed in [40].
5. Cyclic combinational circuits can be more area-efficient than equivalent acyclic circuits. Riedel and Bruck [35] argue convincingly that "*nearly all combinational circuits are best designed with cycles*". For randomly generated multilevel circuits, they found more than about 90% of the cases permitted an average of 15% area improvement, and for standard benchmarks gains in the range between 3%–30%. In [1, 35] it is shown how existing logic synthesis tools (Berkeley SIS package) can be adapted to generate optimized combinational feedback.

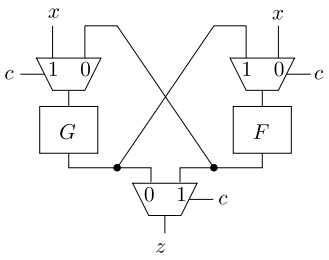


Fig. 1 Sharing of resources leads to a false cycle (Fig. 2 from Malik [23])

In some cases (1 and 3–5), cycles are created intentionally, but with the knowledge that the cycles are false. In other cases (2 and 3), the cycles may have been created inadvertently, and the circuit may or may not be combinational. Regardless of how or why a feedback cycle is created, the only question is whether the resulting circuit is electrically well-behaved. To decide this question it is not sufficient to know the Boolean functions of all gates. As has been observed in [40], well-behavedness is an intensional property. Two circuits may compute the same unique Boolean steady state but one is combinational and the other is not. Whether or not a cyclic circuit is combinational depends equally on the structure of the circuit, where the delays are placed, which electrical delay-model is assumed and the precise operating conditions under which the circuit is exercised.

The intensional nature of the problem gives rise to a whole range of potentially different interpretations of the term 'combinational circuit' which are not necessarily compatible with each other. This point has not received sufficient attention in the literature. Our aim in this article is

- to highlight the fact that there are different classes of 'combinational circuits' associated with different delay models and design styles;
- to characterize in a mathematically precise way the class of cyclic circuits which are known, informally, as *constructive circuits* [40].

Before we give a more detailed summary of our contributions and the structure of this article let us look briefly at the issues involved.

Existing work in the area almost entirely focuses on the *ternary signal model* from which efficient analysis algorithms have been derived. The ternary model $\mathbb{S} = \{0, 1, \bot\}$ is a natural extension of the Boolean two-valued model $\mathbb{B} = \{0, 1\}$ to analyze circuits in the presence of cycles and propagation delays. A third value $\bot$ is added to give a minimum extra capacity for accommodating time-related features of real circuits, without entering the descriptive and algorithmic complexity of a full real-time analysis. Viewed as an extension of classical binary logic the ternary model occurs already in Kleene's work on partial recursive functions [19]. As a three-valued signal algebra the ternary model was introduced by Yoeli and Rinon [51] to analyze static hazards. Eichelberger [13] extended the method to handle general hazards in combinational circuits, as well as races and oscillations in sequential circuits. The theory and application of *ternary simulation* has been developed further by other authors, e.g., at the gate level by Brzozowski, Yoeli, and Seger in [8, 9, 50], or at the transistor level by Bryant [6]. The ternary analysis of cyclic combinational circuits was pioneered by Malik in [23] and subsequently extended in a number of papers such as [1, 10, 31, 34, 40, 43]. In programming languages we mention [4, 38] as examples applying the ternary model for cycle analysis.

Ternary simulation is well-known and long established. However, in so far as it is intended to give an abstraction of electrical circuit behavior, the story remains incomplete.

Little work has been done on relating the three-valued model with actual real-time behavior. In Malik's seminal paper [23], no precise connection with real-time behavior is made and thus the question of what exactly it means for a circuit to be well-behaved is left unanswered. More specifically, it is not clear what the intuitive reading of the third value $\perp$ should be. Does it stand for a particular behavior such as "oscillation" or "transient change"? Is it an "undefined" or "don't care" value, or all of these possibilities? This answer to this question is indeed non-trivial. The paper by Breuer [5] demonstrates some of the paradoxes that arise when the ternary domain is interpreted naively as a domain of signal values to express concrete real-time properties of individual signals. The confusion arises from treating $\perp$ simultaneously as an undetermined stable and a determined unstable value. Algebraically, it is not clear how to make sense of simultaneous equations like $x = \perp$ and $y = \perp$, which would imply $x = y$. For what does it mean that two signals are identical when they are both undefined?

Some results in this direction have been obtained by Brzozowski and Seger [8] for Eichelberger's 2-step simulation and by Shiple [41] for Malik's one-pass algorithm, relating the ternary simulation model to the *up-bounded inertial (*UI*) delay model* (which will be defined below). It is shown that ternary simulation captures, in an approximative sense, the steady state behavior of *complete* UI-networks, i.e., those that have UI-delays associated with *all* gates and *all* wires. This is an important, yet only partial result. It does not say what kind of information ternary simulation computes, in the general case of *non-complete* networks in which only *some* gates and *some* wires have delays.

In this article we will close this semantic gap and resolve the paradoxes of the "third value", concluding that a different delay model must be used. In Sect. 2 we will first observe that for general UI-delay networks ternary simulation is an over-approximation and thus incomplete. In some cases the algorithm may return the unstable value $\perp$ when in fact the circuit signal, under the UI-delay model, stabilizes to a unique Boolean value. This implies that, in contrast to what seems to be suggested by the literature, the UI-delay model and ternary analysis do not quite fit together.

Next, we argue that for non-complete networks, UI-delays have further significant disadvantages: (i) The combinational analysis of UI-steady-state behavior has exponential time complexity where the ternary model is linear. (ii) There exist networks which are combinational under the UI-model but exhibit internal metastable behavior and hence have electrically unbounded response time. (iii) The UI-delay model is not compositional and thus unsuitable for hierarchical circuit analysis. The problems can be traced to the inertiality condition, which leads us to the notion of *up-bounded non-inertial (UN)* delays.

We will then show in Sect. 3, the main part of this article, that for the UN-model a perfect match between the ternary domain and the electrical behavior can be obtained. The key insight is that the more conservative UN-model introduces enough additional signal decoupling so that memory effects and anomalous behavior necessarily come out as oscillations. This will permit the consistent interpretation of the third value $\perp$.

Based on these observations we propose the following general definition: A network (a circuit of Boolean function blocks with a fixed placement of delay nodes) is called *constructive* if it computes unique Boolean state values for all Boolean inputs in bounded time assuming all delay elements are UN-delays. A circuit is *constructive* if the complete network induced by it is constructive.

As far as we are aware this is the first precise definition of a notion of combinational systems that is supported by three equivalent characterizations, analogous to the operational, denotational and axiomatic semantics of programming languages, thus filling in the promises in [40]:

– The *electrical definition*, given in terms of stabilization under the *non-inertial delay model* (Definition 6);
– The *algebraic definition*,[1] based on fixed point computations in the domain of *ternary values* (Corollary 3);
– The *logical definition*, arising from the axiomatic specification of network behavior in *UN-Logic* (Corollary 1).

Establishing these equivalences rigorously is not straightforward. The proof is based on a new constructive modal logic, *UN-Logic*, defined in Sect. 3.1. It acts as a specification language for UN-delay networks. Its *model-theory* adequately captures the *electrical behavior* and its *proof-theory* is related to *ternary algebra*. In this fashion, the problem reduces to a soundness and completeness argument for the fragment of UN-Logic corresponding to network specifications, also called *network theories*. All the essential steps of this are presented in Sects. 3.2 and 3.3, while tedious technicalities, uninteresting for the main story of this article, can be found in [30].

UN-Logic (Sect. 3.1) constitutes the methodological hub of this work. It is a constructive propositional logic equipped with a modal operator $\Diamond_D$ for *bounded stabilization*. It plays the same role for ternary algebra as classical two-valued logic does for Boolean algebra. Constructivity is the key feature that links this logic with ternary algebra: If a network specification satisfies $\Diamond_D s \vee \Diamond_D \overline{s}$ ("*s stabilizes to* 1 *or* 0 in *D* time") then it must already satisfy either $\Diamond_D s$ ("*s stabilizes to* 1 *in D time*") or $\Diamond_D \overline{s}$ ("*s stabilizes to* 0 *in D time*"). Hence, if s does not stabilize to a unique value in bounded time, then s does not stabilize at all, i.e., exhibits oscillation in some execution. Because of this, oscillation becomes an adequate interpretation for the "unknown" value $\perp$ of ternary signal algebra. Moreover, constructivity means that Lamport's conjecture [20] is provable for UN-delay networks, viz. that bounded stabilization requires determinism (Corollary 2).

We conclude in Sect. 4 with a discussion of related work and some open problems arising from the technical developments presented here.

2 When is a circuit "combinational"?

A (circuit) *network* N is an interconnected set of Boolean *function blocks* or *delay elements*. The former may be NOT, AND, OR, NAND gates, complex gates such as multiplexors or PLAs, or combinational subsystems such as decoders, ALUs, etc. All these (network) *components* are linked up through (directed) wires joined at *vertices* $\mathcal{Z} = \{z_1, z_2, \ldots, z_k\}$. The (primary) *input variables* or *input signals* $\mathcal{X} = \{x_1, x_2, \ldots, x_n\} \subseteq \mathcal{Z}$ are all the vertices of N without incoming wires. Every other vertex $\mathcal{Z} \setminus \mathcal{X}$ has its incoming wire connected to the output vertex of exactly one network component. The *state variables* or *state signals* $\mathcal{S} = \{s_1, s_2, \ldots, s_m\} \subseteq \mathcal{Z}$, disjoint from $\mathcal{X}$, are all those vertices whose incoming wire is connected to the output of a delay element. The vertices in $\mathcal{Z} \setminus (\mathcal{X} \cup \mathcal{S})$ are called *combinational*. A network is called *well-formed* if $\mathcal{S}$ forms a feedback vertex set, i.e., every cycle in N contains at least one state vertex. In a *gate state network* the delay elements are connected to all and only the output vertices of function blocks. In a *wire state network* only gate input vertices have delays. Finally, *complete networks* have delays attached to all the input and output vertices of function blocks. While gate-level networks are often complete,

[1] This was called the "semantical definition" in [40].

networks with encapsulated function blocks at higher levels of the design hierarchy may be more adequately modeled as non-complete networks.

With the input and state variables fixed, the idealized Boolean operation of N may be represented by an *excitation* function $S : \mathbb{B}^n \times \mathbb{B}^m \to \mathbb{B}^m$. Since all delays are located in (the updating of) the chosen state variables, S provides an adequate description of the possible state changes of the network, relative to the chosen delay model. Typically, S is given as a vector $S = \langle S_1, S_2, \ldots, S_m \rangle$ of Boolean algebra expressions[2] over $\mathcal{X} \cup \mathcal{S}$, while all other vertices $\mathcal{Z} \setminus (\mathcal{X} \cup \mathcal{S})$ are abstracted out. Given a concrete input $a \in \mathbb{B}^n$ and "current" state $b \in \mathbb{B}^m$, the combined *total state* is denoted $a.b \in \mathbb{B}^n \times \mathbb{B}^m$. The evaluation $S_i(a.b)$ of component S_i on the total state determines the projected "new" value for the state variable s_i. If $S_i(a.b) \neq b_i$ then the state vertex s_i is *excited*, otherwise it is *stable*. If all state vertices are stable, the total state $a.b$ represents a (potential) steady state of the circuit if the input remains unchanged. In *input-output* mode, only a subset of *output* state vertices $\mathcal{O} \subseteq \mathcal{S}$ is considered observable and in *fundamental mode* (see e.g., [8]) all state vertices are observable, i.e., $\mathcal{O} = \mathcal{S}$.

Roughly, a network is *combinational* if it realizes a functional relationship between input and output. For each possible input excitation there is exactly one stable output result at which the circuit will stabilize in bounded time. In particular, the final steady state must not depend on any internal memory. Each time the circuit is used, it will follow the same input-output function table. Combinational circuit analysis then consists of computing this function table along with some characterizing delay bounds. This definition appears easy enough, but when it comes to pinning down this intuition mathematically, things are not quite so clear. A full definition depends on exactly how the circuit is operated by the environment, which real-time delay model is assumed, and where the delays are placed. We may get different results depending on whether we operate in single input change, multiple input change, fundamental, or input-output mode; whether we assume fixed pure delays, up-bounded or bi-bounded, inertial or non-inertial delays; whether we associate delays with every gate and wire or only a subset of them. While these distinctions do not matter for loop-free networks, design styles exploiting combinational feedback must be clear about which interpretation of the term 'combinational' they are using.

The point is that the static excitation function S alone is not sufficient to determine if a particular steady state $a.b$ will be assumed by a network N under a given input a, or how fast N will stabilize to this state. For this we need to enrich Boolean algebra to capture the temporal dimension of the stabilization process. In the following Sects. 2.1 and 2.2 we discuss the two standard approaches, which will bring up the key problem addressed in this article.

2.1 Ternary simulation

If two values are not enough, why not use three? Indeed, the most economic approach is the ternary extension of binary gate modeling, which is based on Kleene's three valued, or ternary, extension $\mathbb{S} = \{0, 1, \perp\}$ [19] of the truth values $\mathbb{B} = \{0, 1\}$. It was used originally by Yoeli, Rinon, Eichelberger, and Brzozowski [9, 13, 51] in order to analyze transient circuit behavior. Depending on context and author the extra value $\perp$ (sometimes also denoted Φ, $\frac{1}{2}$ or X) might stand for some or all of "non-termination", "oscillation", "instability", "transient", "floating", "undefined", "don't care", or similar signal-related features. To keep

[2]Sometimes, we also write vectors without the angled brackets, e.g., $S_1 S_2 \cdots S_m$.

NOT	
0	1
1	0
⊥	⊥

AND	0	1	⊥
0	0	0	0
1	0	1	⊥
⊥	0	⊥	⊥

OR	0	1	⊥
0	0	1	⊥
1	1	1	1
⊥	⊥	1	⊥

XOR	0	1	⊥
0	0	1	⊥
1	1	0	⊥
⊥	⊥	⊥	⊥

Fig. 2 Ternary extensions for the NOT, AND, OR, and XOR functions

matters open, however, it is natural to view $\mathbb{S}$ as a domain of information for specifying elements of $\mathbb{B}$, viz. as "certainly 0", "certainly 1" and "unknown". Under this interpretation $\mathbb{S}$ can be equipped with the obvious *information ordering* $\sqsubseteq$ such that $\mathbf{s} \sqsubseteq \mathbf{t}$ means that $\mathbf{t}$ is more certain, or more defined, than $\mathbf{s}$:

$$0 \sqsubseteq 0, 1 \sqsubseteq 1, \bot \sqsubseteq \bot, \bot \sqsubseteq 0, \quad \text{and} \quad \bot \sqsubseteq 1.$$

Notice that $\bot$ is the bottom element, i.e., maximally uncertain, and that $0, 1$ are mutually incomparable. This corresponds to Scott's ordered Boolean domain, which is also known as the *lifted* set $B_\bot$ of Booleans, familiar from the semantics of programming languages [15, 33].

The domain $\mathbb{S}$ is a complete lower semilattice, i.e., any nonempty subset $\mathbf{X} \subseteq \mathbb{S}$ has a *greatest lower bound* $glb(\mathbf{X})$. Specifically, $glb\{0\} = 0$, $glb\{1\} = 1$ and $glb(\mathbf{X}) = \bot$ for all other non-empty subsets. We can lift the lower semilattice structure from $\mathbb{S}$ to vectors $\mathbb{S}^l$, by point-wise ordering: we put $\mathbf{s} \sqsubseteq \mathbf{t}$ for vectors $\mathbf{t} = \langle \mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_l \rangle$ and $\mathbf{s} = \langle \mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_l \rangle$ if $\mathbf{s}_i \sqsubseteq \mathbf{t}_i$, for all i, and the *glb* of (sets of) vectors $\mathbf{X} \subseteq \mathbb{S}^l$ is computed as

$$glb(\mathbf{X}) =_{\text{df}} \big\langle glb\{\mathbf{s}_1 \mid \mathbf{s} \in \mathbf{X}\}, glb\{\mathbf{s}_2 \mid \mathbf{s} \in \mathbf{X}\}, \ldots, glb\{\mathbf{s}_l \mid \mathbf{s} \in \mathbf{X}\} \big\rangle.$$

Note that $\mathbb{B}^l$ is a sub-domain of $\mathbb{S}^l$. One way to interpret $\mathbb{S}^l$ as an enrichment of $\mathbb{B}^l$ is to consider each $\mathbf{s} \in \mathbb{S}^l$ as a non-empty subset $set(\mathbf{s}) \subseteq \mathbb{B}^l$ of Boolean vectors, viz. $set(\mathbf{s}) =_{df} \{x \mid \mathbf{s} \sqsubseteq x\} \subseteq \mathbb{B}^l$. Both mappings *glb* and *set* are antitonic, i.e., $\mathbf{a} \sqsubseteq \mathbf{b}$ implies $set(\mathbf{b}) \subseteq set(\mathbf{a})$ and $\mathbf{X} \subseteq \mathbf{Y}$ implies $glb(\mathbf{Y}) \sqsubseteq glb(\mathbf{X})$.

For any Boolean function $F : \mathbb{B}^l \to \mathbb{B}$ there exists a natural *ternary extension* $\mathbf{F} : \mathbb{S}^l \to \mathbb{S}$, defined as follows:

$$\mathbf{F}(\mathbf{a}) = glb\big\{F(t) \mid \mathbf{a} \sqsubseteq t,\ t \in \mathbb{B}^n\big\} = glb\big(F\big(set(\mathbf{a})\big)\big).$$

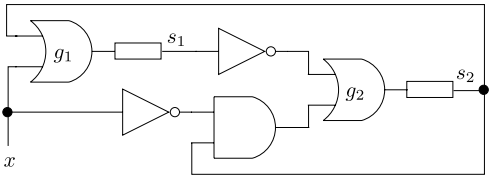
Figure 2 shows the ternary extension for several Boolean functions.

Ternary extensions can be formed for vector-functions, too, in a component-wise manner. The ternary extension of $F : \mathbb{B}^l \to \mathbb{B}^m$ is $\mathbf{F} : \mathbb{S}^l \to \mathbb{S}^m$ such that $\mathbf{F}(\mathbf{y}) = \langle \mathbf{F}_1(\mathbf{y}), \ldots, \mathbf{F}_m(\mathbf{y}) \rangle$. In [39] it is shown how such extensions can be computed.

An important property of the ternary extension $\mathbf{F}$ of a Boolean (vector) function F is that it preserves F on Boolean values, i.e., for all $x \in \mathbb{B}^l$, $\mathbf{F}(x) = F(x)$, and further that it is monotonic, i.e., $\mathbf{a} \sqsubseteq \mathbf{b}$ implies $\mathbf{F}(\mathbf{a}) \sqsubseteq \mathbf{F}(\mathbf{b})$. In other words, if $\mathbf{a}$ is at least as uncertain as $\mathbf{b}$, then the output $\mathbf{F}(\mathbf{a})$ is at least as uncertain as $\mathbf{F}(\mathbf{b})$. By a general result (special form of Knaster-Tarski Theorem, see, e.g., [46]), monotonic functions on finite lower semilattices have fixed points. Specifically, if $\mathbf{F} : \mathbb{S}^n \to \mathbb{S}^n$ is a ternary extension, then

$$\mu\mathbf{y}.\, \mathbf{F}(\mathbf{y}) =_{\text{df}} glb\big\{\mathbf{y} \mid \mathbf{F}(\mathbf{y}) \sqsubseteq \mathbf{y}\big\}$$

Fig. 3 Network N_1



is the least fixed point of $\mathbf{F}$, i.e., $\mathbf{F}(\mu\mathbf{y}.\mathbf{F}(\mathbf{y})) = \mu\mathbf{y}.\mathbf{F}(\mathbf{y})$ and for all $\mathbf{y}'$ such that $\mathbf{F}(\mathbf{y}') = \mathbf{y}'$ we have $\mu\mathbf{y}.\mathbf{F}(\mathbf{y}) \sqsubseteq \mathbf{y}'$. Let $\mathbf{F}^i$ be the i-th iteration of $\mathbf{F}$, i.e., $\mathbf{F}^0(\mathbf{a}) = \mathbf{a}$ and $\mathbf{F}^{i+1}(\mathbf{a}) = \mathbf{F}(\mathbf{F}^i(\mathbf{a}))$. On the finite domain $\mathbb{S}^n$ the fixed point can be computed by an approximation sequence

$$\bot \sqsubseteq \mathbf{F}(\bot) \sqsubseteq \mathbf{F}^2(\bot) \sqsubseteq \mathbf{F}^3(\bot) \sqsubseteq \cdots \sqsubseteq \mathbf{F}^i(\bot) = \mathbf{F}^{i+1}(\bot) = \mu\mathbf{y}.\mathbf{F}(\mathbf{y})$$

in at most $i \leq m$ steps.

The ternary extension $\mathbf{N}$ of a gate-level network N is given by considering N as operating on $\mathbb{S}$ rather than $\mathbb{B}$. Suppose $S : \mathbb{B}^n \times \mathbb{B}^m \to \mathbb{B}^m$ is N's excitation function for inputs $\mathcal{X} = \{x_1, \ldots, x_n\}$ and state vertices $\mathcal{S} = \{s_1, \ldots, s_m\}$. Its ternary extension $\mathbf{S} : \mathbb{S}^n \times \mathbb{S}^m \to \mathbb{S}^m$ can be used to analyze N's combinational behavior[3] (see, e.g., [23, 51]). The least fixed point approximation in variable $\mathbf{y}$ of $\mathbf{S}(a.\mathbf{y})$ for fixed $a \in \mathbb{B}^n \subseteq \mathbb{S}^n$ is a form of symbolic simulation to derive information about N's stabilization behavior under input a (assumed constant throughout). The fixed point $\mu\mathbf{y}.\mathbf{S}(a.\mathbf{y}) \in \mathbb{S}^m$ provides an information-theoretic description of the final stationary state. If $(\mu\mathbf{y}.\mathbf{S}(a.\mathbf{y}))_i = b \in \mathbb{B}$ then state variable s_i will stabilize at Boolean value b, otherwise if $(\mu\mathbf{y}.\mathbf{S}(a.\mathbf{y}))_i = \bot$ the stabilization behavior of s_i is undefined. This least fixed point semantics is applied, implicitly, by Malik [23] and also by Bryant [6] at the transistor level. Ternary simulation gives rise to the following definition of "combinational" behavior:

Definition 1 Let N be a network with state vertices $\mathcal{S}$, outputs $\mathcal{O} \subseteq \mathcal{S}$ and system excitation function $S : \mathbb{B}^n \times \mathbb{B}^m \to \mathbb{B}^m$. Then N is *ternary combinational in input-output mode* if the least fixed point of its ternary extension is Boolean for all output vertices under all Boolean inputs, i.e., if for all $a \in \mathbb{B}^n$ and $s_i \in \mathcal{O}$, we have $(\mu\mathbf{y}.\mathbf{S}(a.\mathbf{y}))_i \in \mathbb{B}$. If $\mathcal{O} = \mathcal{S}$, N is called *ternary combinational in fundamental mode*, or simply *ternary combinational*.

For complete gate-level networks this definition coincides with the term 'combinational' in the sense of Malik [23] where each individual gate is replaced by its ternary extension. Our definition, however, is more generally applicable to networks with complex function blocks, which are not necessarily complete.

Example 1 Consider the network N_1 of Fig. 3 with input $\mathcal{X} = \{x\}$ and state variables $S = \{s_1, s_2\}$, indicated by the delay elements at the outputs of the OR gates g_1 and g_2. The excitation functions are $S_1 = x + s_2$ and $S_2 = \overline{s_1} + \overline{x} \cdot s_2$. The total system function of N_1 is $S = \langle S_1, S_2 \rangle$ with ternary extension $\mathbf{S} : \mathbb{S}^3 \to \mathbb{S}^2$. It is easy to see that for input $x = 1$ the least fixed point is $\mu\mathbf{y}.\mathbf{S}(1.\mathbf{y}) = 10$ and for $x = 0$ we find $\mu\mathbf{y}.\mathbf{S}(0.\mathbf{y}) = \bot\bot$. Since ternary simulation cannot guarantee stable values for s_1 and s_2 when the input is 0, N_1 is not ternary combinational.

[3]The ternary extension can also be used, to some extent, to analyze sequential behavior (as in [8, 9, 13]).

Fig. 4 Delay element

But what does Definition 1 mean for the actual electrical behavior of a circuit network? The property of a network being combinational according to the ternary model does not *per se* imply anything about the network's stationary states because this would have to involve a real-time model of waveforms and propagation delays. The relationship between ternary simulation and a fixed real-time model, which needs to be established separately, has two dimensions:

Correctness: Is it always the case that if $\mu\mathbf{y}.\mathbf{S}(a.\mathbf{y})$ returns a Boolean result b for state variable s_i, then in the real-time model s_i will stabilize to that value b in bounded time?

Exactness: If s_i stabilizes to value b in bounded time (under constant input a) in the real-time model, then does the fixed point $\mu\mathbf{y}.\mathbf{S}(a.\mathbf{y})$ indeed compute b for s_i?

These questions are crucial for the practical interpretation of the results obtained by ternary analysis, but are rarely addressed in the literature. In particular, note that it is exactness that gives real-time meaning to the undefined value $\perp$. Only under exactness does $\perp$ imply that the associated signal actually exhibits anomalous behavior such as oscillation, nondeterminism or unbounded response time. Without exactness $\perp$ is nothing more than "don't know" which includes *all possible* behaviors, even perfect stabilization. In a real-time interpretation for which ternary simulation is exact, however, the fact that $\mu\mathbf{y}.\mathbf{S}(0.\mathbf{y}) = \perp\perp$ in the above Example 1 could be taken as an indication of a real-time anomaly (see the next Sect. 2.2). Such questions of exactness are not addressed in [23] or its extension to timing analysis of cyclic circuits [36].

2.2 Real-time execution

Let us now introduce the realm of time. Real-time models of circuits capture, at a suitable level of abstraction, the dynamics of the electrical processes that unfold when the real physical circuit is running. The trajectory of a signal, or *history*, measured in terms of voltage, current, etc., would be described by a continuous function $s : \mathbb{R}^+ \to \mathbb{R}$, governed by differential equations. At gate level, however, the standard model of an asynchronous digital circuit typically abstracts from the continuous values and assumes histories are binary $s : \mathbb{R}^+ \to \mathbb{B}$ and specified by (timed) Boolean equations. Thus, histories $s(t)$ vary in real-time but are capable of instantaneous changes from 0 to 1 and from 1 to 0. It is mathematically expedient to assume that histories are *right-continuous*, i.e., if $s(t) = \alpha$ then $s(t)$ is constant α throughout some non-empty closed-open interval $[t, t+\epsilon)$ to the right of t. For any signal s, times $t_1 \in \mathbb{R}$, $t_2 \in \mathbb{R}^+_\infty = \mathbb{R}^+ \cup \{+\infty\}$ and value $\alpha \in \mathbb{B}$ let us write $s[t_1, t_2) = \alpha$ to abbreviate $\forall t_1 \leq t < t_2 \Rightarrow s(t) = \alpha$. Right continuity, then, is the same as $\forall t.\, \exists \epsilon > 0.\, s[t, t+\epsilon) = s(t)$. Moreover, signals must be *non-Zeno*, i.e., they cannot make an infinite number of changes within a bounded time interval. These two conditions are reflections of the underlying (assumed) continuous physics. Notice that non-Zeno-ness implies that for every t there exists $\delta > 0$ such that $s[t-\delta, t) = \alpha$ for some $\alpha \in \mathbb{B}$. These assumptions are consistent with the theory of *Timed Boolean Functions* proposed by Lam and Brayton in [21].

The real-time model of a gate network is a combination of its Boolean excitation functions and the effect of the delay elements associated with the chosen state vertices. A delay element has an input $S(t)$ and an output $s(t)$, as depicted in Fig. 4. In network terms, s represents a state variable, and S its excitation function.

A delay element establishes a temporal relationship or transformation between its input and output waveforms. If $\mathsf{DEL} \subseteq (\mathbb{R}^+ \to \mathbb{B}) \times (\mathbb{R}^+ \to \mathbb{B})$ denotes this relation then we could state this as $\mathsf{DEL}(S, s)$. In general, DEL is non-functional. For any given excitation waveform S there may be many possible outputs s such that $\mathsf{DEL}(S, s)$, e.g., relating to different (internal) initializations and different signal propagation times though the delay element.[4]

Now we move from the behavior of a single delay element to the behavior of a network containing multiple delay elements. For state variable, $s_i(t)$ is the value of s_i at time t, $x(t)\centerdot s(t)$ gives the value at time t of the total state extended by the input and $S_i(x(t)\centerdot s(t))$ is the value of the corresponding excitation at time t. From now on, $S_i(x\centerdot s)$ is an excitation waveform rather than a single excitation value. A DEL-history(N, a), defined next, captures the behavior of a network as it evolves over time in response to an input a when all delays are modeled by DEL. Since signal propagation through delay elements is non-deterministic, there may be an infinite number of DEL-histories for a given input a.

Definition 2 A *DEL-history*(N, a) of a network N for delay model DEL and input value $a \in \mathbb{B}^n$ is a total state trajectory $h(t) = x(t)\centerdot s(t)$, which is right-continuous and non-Zeno such that for all $t \geq 0$, $x(t) = a$ and for all $s_i \in \mathcal{S}$ the state update and excitation conditions $\mathsf{DEL}(S_i(x\centerdot s), s_i)$ are satisfied.

For a given delay model DEL, we can now formalize the real-time interpretation according to which a network is combinational if its stable behavior implements a *functional* input-output relationship (in particular it must not depend on internal memory). In other words, whenever the input is held constant at some value a the system will stabilize in bounded time to a resulting final state that is uniquely determined by this input value.

Definition 3 (DEL-combinational) A network N is *DEL-combinational in input-output mode* if for all input vectors a and output state signals $s_i \in \mathcal{O} \subseteq \mathcal{S}$, there exists a stabilization delay $\delta \geq 0$ and final state $b \in \mathbb{B}$ such that in all $x(t)\centerdot s(t) \in \mathsf{DEL}\text{-history}(N, a)$ we have $s_i(t) = b$ for all $t \geq \delta$. If $\mathcal{O} = \mathcal{S}$, then N is called *DEL-combinational in fundamental mode* or simply *DEL-combinational*.

Various delay models have been introduced in the literature, ranging from fixed ideal delay to bi-bounded inertial delay [8]. The delay model that has been discussed most suggestively in connection with ternary simulation is the *up-bounded inertial* (UI) delay model [8] introduced below in Definition 4. This model captures a superset of the behaviors possible in the fixed ideal delay and bi-bounded inertial delay models, and thus is more conservative than either of them. Considering such a wide and conservative range of behaviors is appropriate when determining the function of circuits with cycles. We will see, though, that it is not quite conservative enough.

2.2.1 The inertial delay model

The inertial delay model is one possible way of generating the dynamic stabilization behavior of networks from their Boolean excitation functions.

[4]The non-determinism in the delay-model DEL accounts for low-level physical parameters which influence the signal propagation delay electrically but can neither be controlled nor kept track of by the discrete model.

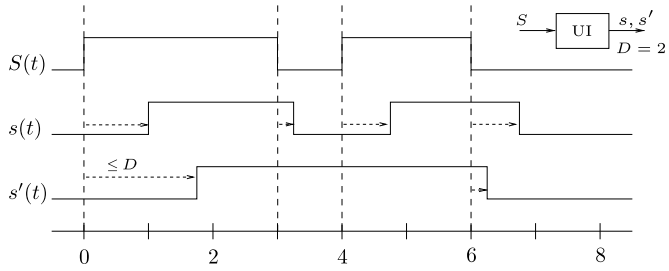


Fig. 5 Up-bounded inertial delay waveforms

Definition 4 (Brzozowski and Seger [8]) In the *up-bounded inertial* delay model UI(S, s), the delay is bounded from above by a parameter $D > 0$ and the following two properties must be satisfied:

- **Inertiality:** *If s changes, then it must have been unstable.*
  Formally, if $s(t)$ changes from α to $\overline{\alpha}$ at time τ, then there exists $\delta > 0$ such that $S[\tau - \delta, \tau) = \overline{\alpha}$.
- **Propagation:** *The output s cannot be unstable for more than D units of time without changing.*
  Formally, if $S[t_1, t_2) = \alpha$ for $t_2 > t_1 + D$, then there exists a time $t \in [t_1, t_2)$, such that $s[t, t_2) = \alpha$. (Note that this implies that the δ in the Inertiality Condition cannot be greater than D.)

Intuitively, if an input pulse is at least D units of time, then by Propagation the delay's output must respond within D time. If an input pulse is less than D, then the output may or may not respond. If the input returns to its original value before the output changes, then by Inertiality the delay element is stable again. Note that D cannot be zero; this is to avoid zero-delay loops in the network.

Example 2 To illustrate Definition 4, Fig. 5 shows two possible responses to an input waveform, where $D = 2$. Observe that in the response $s'(t)$ the second input change of $S(t)$, which happens at time 3, is not propagated by the delay. The input $S(t)$ returns to the old value at time 4 before the output $s'(t)$ has reacted. Inertiality then requires that the input must first switch again (at time 6) in order for the output to make any change at all.

Example 3 A possible UI-history of N_1 (Example 1) for input $x = 0$ and initial state $S(0) = s_1(0)s_2(0) = 10$ is seen in Fig. 6. The two delay elements have $D_1 = 1$, $D_2 = 3$. One can verify that these waveforms are consistent with Definition 4 for UI-delays (for the specified delay bounds) and the excitation functions: Let S_i represents the excitation signal at the input of the respective delay. The depicted history $x(t)\centerdot s_1(t)s_2(t)$ satisfies the four conditions

$$\begin{aligned} &\mathrm{UI}(S_1, s_1) \qquad S_1 = x + s_2 \\ &\mathrm{UI}(S_2, s_2) \qquad S_2 = \overline{s_1} + \overline{x}\centerdot s_2. \end{aligned}$$

For example, in the interval $[1, 3.5)$ the state is $s_1s_2 = 00$ and the excitation functions yield $S_1S_2 = 01$. In this interval the delay element for s_1 is stable while that driving s_2 is excited. Within $D_2 = 3$ time s_2 changes, viz. at 3.5 time. Then, between $[3.5, 4.25)$ we have $s_1s_2 =$

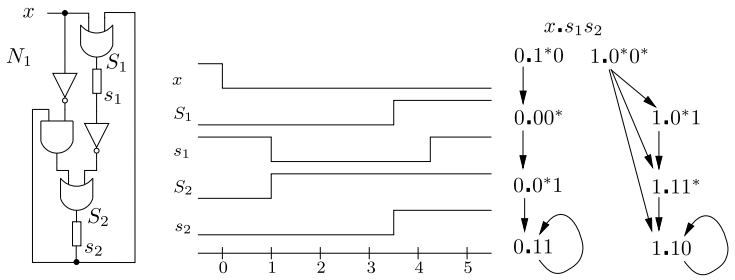


Fig. 6 UI-history$(N_1, 0)$, timed (*center*) and untimed (*right*). The delays are $D_1 = 1$, $D_2 = 3$

01 and $S_1S_2 = 11$. Now the signal s_2 is stable and s_1 excited. The delay parameter is $D_1 = 1$ and indeed this unstable period $[3.5, 4.25)$ lasts only $0.75 < D_1$ units. When s_2 finally switches, at time 4.25, the state becomes $s_1s_2 = 11$ in which both delays are stable. We have reached a total stable state in which the network must remain forever.

With every UI-history one can associate a corresponding untimed history giving the sequence of states through which the network passes. The untimed history for $x\mathbf{.}s_1s_2$ corresponding to the waveform (starting at $t = 0$) just described is $0\mathbf{.}10$–$0\mathbf{.}00$–$0\mathbf{.}01$–$0\mathbf{.}11$. It is depicted as the left one of the two transition graphs seen on the right of the timing diagram in Fig. 6. The other one corresponds to the untimed history that evolves from the initial configuration $1\mathbf{.}00$ (not shown as a timing diagram). In every state each unstable variable is labeled by an asterisk. The self-loops in $0\mathbf{.}11$ and $1\mathbf{.}10$ indicate that these states are stable. These untimed transition diagrams are the *General Multiple Winner* model [8] representing all possible UI-delay executions of N_1.

From the transition graphs on the right of Fig. 6 we can see that in whatever total state the network is started, it must eventually (in bounded time) reach a unique stable state. Following the possible trajectories we notice that for $x = 0$ it must eventually settle in the stable total state $0\mathbf{.}11$ possibly passing through the intermediate states $0\mathbf{.}1^*0$, $0\mathbf{.}00^*$, $0\mathbf{.}0^*1$, in this order. For input $x = 1$ the network must end up in $1\mathbf{.}10$, regardless whether it was started in $1\mathbf{.}0^*0^*$, $1\mathbf{.}0^*1$, $1\mathbf{.}11^*$, or $1\mathbf{.}10$. Thus, the final stable state is uniquely determined by the input and computed in bounded time.

In Example 1 we have seen that network N_1 is not ternary combinational, i.e., not combinational in the sense of Malik. Recall that the ternary fixed point $\mu\mathbf{y}.\mathbf{S}(0\mathbf{.}\mathbf{y}) = \perp\perp$ suggests that both state variables are undefined for input $a = 0$. If ternary simulation were complete for the UI-delay model, this would require the existence of UI-histories in which neither s_1 nor s_2 ever stabilizes in bounded time under this input. This, however, is not the case. Example 3 shows that N_1 is UI-combinational according to Definition 3. Obviously, ternary simulation is an over-approximation which does not detect that N_1 stabilizes under the UI-delay model.

In view of this mismatch it appears odd that the UI-delay model should be used to motivate ternary simulation, as done in [8]. The UI semantics and the ternary semantics give rise to different classes of combinational networks. The relationship between them is at best a partial one. In fact, existing exactness results of Brzozowski and Seger [8, Theorems 7.1, 7.2] for ternary simulation with respect to UI-delays are restricted to complete networks, i.e., in which *all* gates and *all* connection lines are subjected to UI-delays. Obviously, network N_1 (Fig. 3) is not complete. For instance, none of the feedback lines after the fork at s_2 have delays.

In order to get exactness, we must change one of the two models. Either we refine ternary simulation to make it more expressive, or use a different, more conservative, timing model. There are several compelling arguments in favor of the latter approach. First, any extension of the ternary model to analyze arbitrary networks under UI-delays is bound to be computationally expensive while efficient techniques are available for the ternary model. Second, the inertiality of UI-combinational networks makes them electrically problematic because of internal metastability, which jeopardizes the boundedness assumption on response time. Third, the UI-model is intrinsically non-compositional, making hierarchical system analysis infeasible.

In this article we resolve the match between ternary simulation and the delay model at the other end. Ternary simulation is an efficient, widely-used, and highly successful analysis technique. Since it surely computes some real-time information we can try to adjust the delay model so it fits the bill. Indeed this is possible as we will now briefly sketch. The main contribution of this article is to prove this rigorously in Sect. 3.

2.2.2 *The non-inertial delay model*

What prevents the UI model from being an adequate interpretation of the information maintained and computed by ternary simulation? The issue with non-complete networks like N_1 of Fig. 3 is that the inertial delay model is too strong in the sense that it enforces too much synchronization between signals. It does not leave enough room for behaviors (histories) to implement the ternary prediction.

Example 4 Why does the network N_1 stabilize rather than oscillate as the ternary simulation outcome predicts? The heart of the matter is the strong latching behavior of UI-delay elements. Since 11 is a fixed point of the excitation function, i.e., $S(11) = \langle 0+1, \overline{1} + (\overline{0}\cdot 1)\rangle = 11$, the system latches up in state 11 instantaneously at the very moment the two state signals s_1, s_2 assume values 11 (for whatever reason) for the first time. More generally, because of *inertiality*, if the output of a delay element s_i changes at all, it can only change to the value determined by the excitation function S_i evaluated in an ϵ-interval immediately before the change. In this way, the first change of the system *must* result in one of the states from $\{S(00), S(11), S(10), S(01)\} = \{00, 01, 11\}$, the second in some state from $\{S(00), S(01), S(11)\} = \{01, 11\}$, and finally, with the last change $\{S(01), S(11)\} = \{11\}$ we *must* reach the unique stable state 11.

Without the inertiality condition the story would be quite a different one. Suppose we permit delay elements to have a certain amount of internal memory; say they are all composed of two cascaded UI-delays as seen on the left of Fig. 7. These are no longer inertial and the oscillatory behavior seen on the right of Fig. 7 for input $x = 0$ is then perfectly permissible for our network N_1.

The Boolean system functions $S_1 = x + s_2$ and $S_2 = \overline{s_1} + \overline{x}\cdot s_2$ are satisfied at all times and the observed input-output waveforms at the delays, viz. from S_1 to s_1 and from S_2 to s_2, are possible behaviors assuming 2-stage UI-delays as in Fig. 7. In the system trajectory shown the state vector $s_1 s_2$ oscillates through all four possible states 10–11–01–00, in this order. This supports exactly the prediction $\perp\perp = glb\{10, 11, 01, 00\}$ for the stationary state made by ternary simulation.

The new definition is obtained simply by dropping the requirement of inertiality from UI-delays (cf. Definition 4):

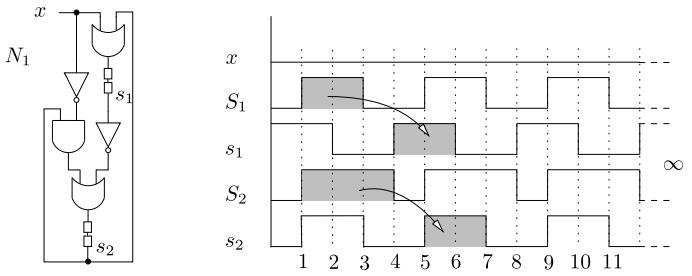


Fig. 7 N_1 exhibits oscillation for non-inertial delays

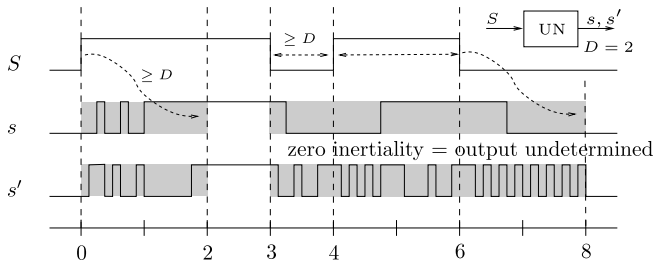


Fig. 8 Up-bounded non-inertial delay waveforms ($D = 2$)

Definition 5 In the *up-bounded non-inertial* delay model (UN-delay), the delay is bounded from above by a parameter $D > 0$ and the following property must be satisfied:

– **Propagation:** *s cannot be unstable for more than D units of time without changing.*
Formally, if $S[t_1, t_2) = \alpha$ for $t_2 > t_1 + D$, then there exists a time $t \in [t_1, t_2)$, such that $s[t, t_2) = \alpha$.

We write $\mathrm{UN}_D(S, s)$ to express that there is an UN-delay element with delay parameter D connecting S and s.

Example 5 Figure 8 shows two possible UN responses to an input waveform S, where $D = 2$. There are two intervals during which input S is stable for more than $D = 2$ units of time, viz. $S[0, 3) = 1$ and $S[6, \infty) = 0$. By the Propagation condition, this means that the output must satisfy $s[2, 3) = 1$, $s[8, \infty) = 0$. These are the only intervals in which the output is driven by the input. The UN-delay, in contrast to UI, requires nothing for the times in between, i.e., the shaded intervals in Fig. 8. Hence, the behavior of the output is completely unconstrained during $[0, 2)$ and $[3, 8)$. This amounts to zero inertiality. At the very moment the input changes, e.g., at time 3, it gives up control. The output becomes completely undetermined until the input has been stable for more than D time again, which happens at time 8. During $[3, 8)$ the output may behave arbitrarily as indicated in response s'.

In accordance with Definition 2, a UN-history(N, a) of a network N is a total (right-continuous, non-Zeno) state trajectory $h(t)$ in which the input is held constant at a and state excitation is governed by the UN-delay model. Since UN-delays admit more real-time behaviors compared to UI-delays, the corresponding notion of a combinational network ("all UN-histories stabilize") is more demanding. Every UN-combinational network is UI-

combinational but not vice versa as the example N_1 demonstrates. This additional stringency in the real-time requirements brings about several major advantages:

- As we will show in Sect. 3, ternary simulation is not only correct but also exact for UN-delays on *arbitrary* networks. In particular, the question if a network stabilizes for fixed input is decidable in linear time.
- The more conservative UN-delay model detects potential metastability problems, which the UI-model ignores (see also Sect. 3.2, p. 309). In [30, Fig. 9, Sect. 2.3.2] an "improved" RS-latch is presented which is non-deterministic but UI-combinational, i.e., all its UI-histories enter a stable state in *bounded* time. The metastability problems of the "improved" RS-latch are masked out following an idea of Pěchouček [32]. Pěchouček's trick works due to the inertiality of UI-delays [30, Sect. 2.3.2]. In contrast, under the UN-model the "improved" RS-latch still has oscillations, whence it is not classified UN-combinational. This is a more sensible conclusion from the electrical point of view, considering results like those of Marino [24] which predict that all binary-state electrical systems must show metastability and thus do not have guaranteed bounded response time (under arbitrary input conditions). Hence, boundedness for UI-combinational networks is guaranteed (empirically verified, see, e.g., [17]) only up to error probabilities while for UN-combinational networks it seems electrically sound.
- The UN-delay model is compositional. The composition of a UN-delay between x and y of delay δ_1 and another between y and z of value δ_2 is equivalent to a single UN-delay of value $\delta_1 + \delta_2$ between x and z. This is not the case for UI-delays as our example N_1 shows (Fig. 6 versus Fig. 7).
- The UN-delay accounts for non-deterministic initialization. During the initial time interval $[0, D)$ the output s of $\mathrm{UN}_D(S, s)$ with parameter D is unconstrained. Hence, a UN-combinational network does not depend on initialization. This reflects much better the idea that the output of a combinational system must be driven completely by the inputs and cannot rely on any internal memory effect (or internal arbitration, for that matter).

All this provides convincing evidence for the important role of the non-inertial delay model for a robust notion of combinational systems, suggesting the following definition:

Definition 6 (Electrical characterization of constructive networks) A (well-formed) network N is *constructive* if N is UN-combinational, i.e., state signals $\mathcal{S} = \{s_1, \ldots, s_m\}$ stabilize in bounded time under the UN-delay model, for all input vectors. Formally, for all $i = 1, \ldots, m$ and $a \in \mathbb{B}^n$ there exists $\tau \in \mathbb{R}^+$ and $\alpha \in \mathbb{B}$ such that for all $h \in \text{UN-history}(N, a)$, $h_i[\tau, \infty) = \alpha$.

The next and main section of this article will establish the close correspondence between UN-delays and ternary simulation. It will provide equivalent logical and algebraic characterizations of constructive networks and further justify the term 'constructive' as used here and originally introduced in the semantics of Esterel [4, 40].

3 Correctness and exactness of ternary simulation

As our first step it will be useful to reformulate the notion of constructivity in terms of the *steady state* behavior of a network, that is, after the transients have died off. We are interested in the set of states that can occur arbitrarily late in the history of the network. Formally, we say a state s is *UN-recurrent* for input a, or simply *recurrent*, if for every time τ there

is a history $h \in \text{UN-history}(N, a)$ and $t \geq \tau$ such that $h(t) = s$. A state may be recurrent because there is a single history in which s appears infinitely often or because it appears arbitrarily late in different histories. Let $\text{UN-recurrent}(N, a)$ be the set of recurrent states for a. The complement of $\text{UN-recurrent}(N, a)$ is the set of *transient states*, abbreviated $\text{UN-transient}(N, a)$.

The notions of transient and recurrent states can be used to express stability. One can show that for a well-formed network N the set $\text{UN-recurrent}(N, a)$ is always non-empty. Then, N *stabilizes* in state s under input a iff every state different from s is transient, i.e., $\mathcal{S} \setminus \{s\} = \text{UN-transient}(N, a)$ or, equivalently, $\text{UN-recurrent}(N, a) = \{s\}$.

Proposition 1 *A network N is constructive iff the set* $\text{UN-recurrent}(N, a)$ *has exactly one element for all inputs a.*

The main result of this section is that ternary simulation is a correct and exact computation of $\text{UN-recurrent}(N, a)$. Before we go into the details of the proof, presented in Sects. 3.1–3.3, let us give an overview of the argument, summed up in Fig. 9.

Suppose we are given a network N with state excitation function $S : \mathbb{B}^n \times \mathbb{B}^m \to \mathbb{B}^m$. Our ultimate claim is that the ternary fixed point $[\![N]\!](a) =_{\text{df}} \mu\mathbf{y}.\mathbf{S}(a.\mathbf{y}) \in \mathbb{S}^m$ computes the set of recurrent states, i.e.,

$$set\big([\![N]\!](a)\big) = \text{UN-recurrent}(N, a). \tag{1}$$

In other words, the ternary state vector of the fixed point, when interpreted as a region of the state space, is precisely the set of recurrent states. The $\supseteq$-direction of (1) contains the *correctness* of ternary simulation as it means that the ternary approximation does not miss any recurrent states. The $\subseteq$-direction of (1) states *exactness*, i.e., that the ternary approximation does not contain superfluous, transient elements.

In view of Proposition 1 the equation (1) implies that N is constructive iff N is ternary combinational since $set([\![N]\!](a))$ is a singleton set iff $[\![N]\!](a) \neq \bot$. This is our *algebraic characterization* of constructive systems (Corollary 3 below).

We will prove (1) by linking the algebraic description $[\![N]\!](a)$ and the electrical behavior $\text{UN-recurrent}(N, a)$ through a logical formalism and an associated *logical characterization* of constructiveness. In doing so, we exploit the two-sided nature of logics which provide at the same time proof-theoretic deduction systems and model-theoretic specification languages, coupled through soundness and completeness theorems. Roughly speaking, we will arrange things so that the deduction calculus for our logic emulates ternary simulation while the semantic interpretation of formulas captures the electrical behavior of a network.

What logic formalism should we choose? Since steady-state analysis does not require full information about signal waveforms, the required level of expressiveness lies somewhere properly between classical propositional logic (Boolean algebra) and predicate logic (Timed Boolean Functions [21]). The former is too weak as it cannot express timing and the latter is too strong because it also captures transient behaviors of combinational systems, and even the sequential behavior of flip-flops, which is unnecessary for the problem at hand. To fit the bill, we introduce a dedicated propositional modal logic, dubbed *UN-Logic*. Our proof strategy is illustrated in Fig. 9, which we will now describe in more detail.

Starting with Sect. 3.1 we extend classical propositional logic by a modal operator $\Diamond_D$ that relaxes truth just enough to faithfully represent propagation delays: for any *region* $R \subseteq \mathbb{B}^m$ in the state space and delay subscript $D \in \mathbb{R}$, $\Diamond_D R$ says that R is reached "within bounded delay D". Then, we show that the behavior of a network N under input a can be

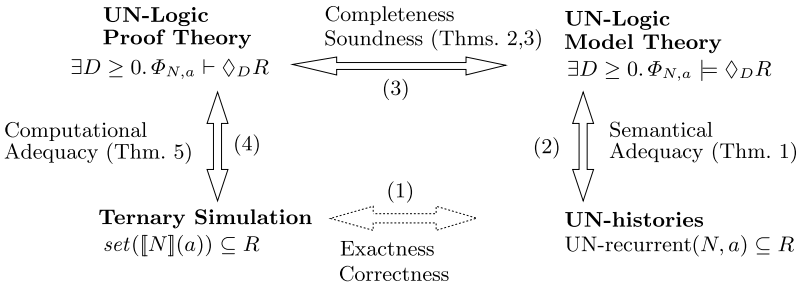


Fig. 9 Overview of the proof strategy

captured by a finite set of formulas $\Phi_{N,a}$ such that a history h satisfies all formulas in $\Phi_{N,a}$ iff $h \in \text{UN-history}(N, a)$. This is the *Semantical Adequacy Theorem* (Theorem 1) derived in Sect. 3.1. Defining *semantical consequence* $\Phi_{N,a} \models \psi$ to mean that all histories satisfying $\Phi_{N,a}$ also satisfy formula ψ, Semantical Adequacy implies that for all regions R

$$\text{UN-recurrent}(N, a) \subseteq R \quad \text{iff } \exists D \geq 0.\, \Phi_{N,a} \models \Diamond_D R. \tag{2}$$

In particular, taking $R = \{s\}$, we find that a network N stabilizes to state s in bounded time iff $\Phi_{N,a} \models \Diamond_D\{s\}$ for some $D \geq 0$. Note that (2) is equivalent to saying that UN-recurrent(N, a) is the smallest region such that $\Phi_{N,a} \models \Diamond_D R$ for any $D \geq 0$. This yields the model-theoretic formulation of our *logical characterization* of constructive networks (Corollary 1): A network N is constructive if for every input a there is a unique state s such that $\Phi_{N,a} \models \Diamond_D\{s\}$ for $D \geq 0$ and $\Phi_{N,a} \not\models \Diamond_D\emptyset$ for no $D \geq 0$.

The next and crucial step, taken in Sect. 3.2, will be to set up a proof system for deriving formal entailment $\vdash$ between modal propositions and to prove *Soundness and Completeness Theorems* (Theorems 2, 3), from which it follows, for all regions R and delay bounds D, that

$$\Phi_{N,a} \models \Diamond_D R \quad \text{iff } \Phi_{N,a} \vdash \Diamond_D R. \tag{3}$$

In other words, $\Diamond_D R$ is semantically valid in all histories of N iff $\Diamond_D R$ is derivable from the specification $\Phi_{N,a}$ in the deduction system for UN-Logic. This translates the semantic condition $\models$ into a derivability statement $\vdash$ and thus reduces constructiveness of a network N to a question on formal derivability which can be computed algorithmically. This is the proof-theoretic formulation of our *logical characterization* of constructive networks.

It turns out that the decision procedure for axiomatizations of the form $\Phi_{N,a}$ is particularly simple because $\Phi_{N,a}$ is a Horn theory. Indeed, in Sect. 3.3 we generalize the well-known relationship between Horn-clause theories and fixed points to prove that for circuit specifications of the shape $\Phi_{N,a}$, derivability of formulas such as $\Diamond_D R$ is equivalent to timed ternary simulation. Our *Computational Adequacy Theorem* (Theorem 5) essentially states that

$$set\big([\![N]\!](a)\big) \subseteq R \quad \text{iff } \exists D \geq 0.\, \Phi_{N,a} \vdash \Diamond_D R. \tag{4}$$

This is the same as saying that $set([\![N]\!](a))$ is the smallest region R such that $\Phi_{N,a} \vdash \Diamond_D R$ for some $D \geq 0$, i.e., the smallest region that N provably reaches in bounded time, in UN-Logic.

As indicated by the dotted double arrow in Fig. 9, combining (2), (3) and (4), we obtain (1) as desired: Since $set([\![N]\!](a)) \subseteq set([\![N]\!](a))$ the ($\Rightarrow$)-direction of (4) gives $\exists D \geq 0.\ \Phi_{N,a} \vdash \Diamond_D set([\![N]\!](a))$. From here, the ($\Leftarrow$)-direction of (3) yields $\exists D \geq 0.\ \Phi_{N,a} \models \Diamond_D set([\![N]\!](a))$ and finally ($\Leftarrow$) of (2) gives us UN-recurrent$(N,a) \subseteq set([\![N]\!](a))$. The converse $set([\![N]\!](a)) \subseteq$ UN-recurrent(N,a) follows symmetrically, using (2)–(4) in their other directions, starting from the trivial observation that UN-recurrent$(N,a) \subseteq$ UN-recurrent(N,a).

Readers not interested in the details may now skip the rest of this section and continue with the Conclusion Sect. 4.

3.1 UN-logic: a specification language for UN-delay networks

UN-Logic is a propositional temporal logic over time intervals that captures the temporal dimension of stabilization by a modal operator $\Diamond_D$ such that $\Diamond_D \phi$ means "within some delay bounded by D, ϕ becomes true". The logic is a multi-modal ("multi" because of the delay indices of $\Diamond$) extension of Boolean algebra with the following syntax:

$$\phi ::= R \mid \neg\phi \mid \phi \supset \phi \mid \phi \equiv \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \Diamond_D \phi,$$

where $R \subseteq \mathbb{B}^{\mathcal{Z}}$, for a set of network vertices $\mathcal{Z} = \{z_1, z_2, \ldots, z_k\}$, ranges over subsets of the total state space, called *regions*, and $D \geq 0$ over non-negative real numbers.

Regions R are constraints on the state of a network specified as Boolean equations in variables $\mathcal{Z}$. For instance, R might be the equation $z_4 = x_1 \cdot y_2 + z_5$, where $x_1, y_2, z_4, z_5 \in \mathcal{Z}$ refer to specific vertices in the network. Then, $\Diamond_D(z_4 = x_1 \cdot y_2 + z_5)$ stipulates that the system must enter the state space region characterized by $z_4 = x_1 \cdot y_2 + z_5$ within D time units. Instead of the equation we could use the Boolean expression $z_4 \cdot (x_1 \cdot y_2 + z_5) + \overline{z_4} \cdot \overline{x_1 \cdot y_2 + z_5}$ which amounts to the same condition. A formal expression like $z_4 = x_1 \cdot y_2 + z_5$ will be identified with the underlying set $\{c \in \mathbb{B}^{\mathcal{Z}} \mid c \textit{ satisfies } z_4 = x_1 \cdot y_2 + z_5\}$ of total state vectors. In this way, the semantics of Boolean algebra is built into the notion of regions. This is convenient as it means our formalization does not need to bother about axiomatizing Boolean algebra, which is well understood. In a practical implementation one would use BDDs or SAT techniques to handle the Boolean part.

Definition 7 (Validity) Formulas are interpreted relative to a given history $h : \mathbb{R}^+ \to \mathbb{B}^{\mathcal{Z}}$ and a non-empty open-closed interval $[s,t)$ demarcated by time points $s < t$, where t may be ∞. We write $h[s,t) \models \phi$ to state that history h *satisfies* (or *validates*) ϕ in the interval $[s,t)$ according to the following inductive clauses:

$h[s,t) \models R$	iff h moves fully inside R during $[s,t)$, i.e.,$\forall \tau \in [s,t).\ h(\tau) \in R$;
$h[s,t) \models \neg\phi$	iff h does not satisfy ϕ in any sub-interval $[s',t')$ of $[s,t)$, i.e., $\forall s', t'.\ s \leq s' < t' \leq t \ \Rightarrow\ h[s',t') \not\models \phi$;
$h[s,t) \models \phi \wedge \psi$	iff $h[s,t) \models \phi$ and $h[s,t) \models \psi$;
$h[s,t) \models \phi \vee \psi$	iff $h[s,t) \models \phi$ or $h[s,t) \models \psi$;
$h[s,t) \models \phi \supset \psi$	iff in every sub-interval $[s',t')$ of $[s,t)$ in which h satisfies ϕ, it also satisfies ψ, i.e., $\forall s', t'.\ (s \leq s' < t' \leq t\ \wedge\ h[s',t') \models \phi)\ \Rightarrow\ h[s',t') \models \psi$;
$h[s,t) \models \phi \equiv \psi$	iff h satisfies ϕ and ψ in the same sub-intervals, i.e., $\forall s', t'.\ s \leq s' < t' \leq t \Rightarrow (h[s',t') \models \phi\ \Leftrightarrow\ h[s',t') \models \psi)$;
$h[s,t) \models \Diamond_D \phi$	iff assuming $[s,t)$ is long enough, h satisfies ϕ with delay D, i.e., $s + D < t\ \Rightarrow\ h[s+D,t) \models \phi$.

We say that h *validates* ϕ, or h is a *model* of ϕ, and write $h \models \phi$ if $h\,I \models \phi$ for *all* (non-empty) closed-open intervals I. The semantics is set up so that $h\,I \models \phi$ implies $h\,J \models \phi$ for all sub-intervals $J \subseteq I$, as one can show by induction on formulas. Hence, $h \models \phi$ is equivalent to $h[0,\infty) \models \phi$. A history h is a *model* of a formula set Φ in some interval $[s,t)$, written $h[s,t) \models \Phi$, if $h[s,t) \models \phi$ for all $\phi \in \Phi$. The semantic consequence $\Phi \models \psi$ between formula sets Φ and formulas ψ, too, is defined locally to intervals: Every history h that is a model of all $\phi \in \Phi$ in some interval $[s,t)$ also validates ψ in $[s,t)$.

Let us make a couple of simple but useful observations. The empty region $\emptyset \subseteq \mathbb{B}^{\mathcal{Z}}$, representable by the Boolean expression 0, corresponds to the proposition *false* and the full space $\mathbb{B}^{\mathcal{Z}}$, represented by 1, corresponds to *true*. We always have $h[s,t) \models 1$ and never $h[s,t) \models 0$. Once we have *false* we can define negation $\neg\phi$ as an abbreviation of the implication $\phi \supset \mathit{false}$. One can show that the logical negation $\neg R$ of a region R is equivalent to the complement region $\overline{R}$. So, for instance $\neg(x \cdot \overline{y})$ has the same meaning as $\overline{x} + y$. Similarly, the other two propositional operators $\supset$ and $\wedge$ are representable on regions: $R_1 \wedge R_2$ is the same as $R_1 \cap R_2$ (as sets) or $R_1 \cdot R_2$ (as Boolean expressions) and $R_1 \supset R_2$ the same as $\overline{R_1} \cup R_2$ or $\overline{R} + R_2$. Equality $R_1 = R_2$ on regions or Boolean expressions is the same as bi-implication $R_1 \equiv R_2$, where the latter, as a logical equivalence, may be seen as an abbreviation for $(R_1 \supset R_2) \wedge (R_2 \supset R_1)$. Furthermore, a formula ϕ is equivalent to $\phi \equiv 1$ and $\neg\phi$ is equivalent to $\phi \equiv 0$.

The observations above essentially imply that the fragment consisting of operators $\{R, \neg, \wedge, \supset, \equiv\}$ is a Boolean algebra. The two operators that are *not expressible* in Boolean algebra are $\vee$ and $\Diamond_D$ for they are related to stabilization behavior. For instance, $h \models R_1 \vee R_2$ specifies that all states assumed by h are contained in R_1 or all states are included in R_2. This is different from saying that all states are in $R_1 \cup R_2$. In the first case the decision for R_1 or R_2 has to be stable throughout the history, in the second case it is pointwise and can switch with every point in time. Obviously, $R_1 \vee R_2$ is stronger than $R_1 \cup R_2$, i.e., we have $h \models R_1 \vee R_2$ implies $h \models R_1 \cup R_2$ but not vice versa. The other operator providing additional expressiveness is the modality. There is no region R' that would represent $\Diamond_D R$ ("after delay D, R holds"), except where $D = 0$. It is obvious from the definition that $\Diamond_0 R$ is in fact the same as R, which is a useful special case. We will call $\Diamond_D R$ for any D and R a *timed* or *modalized* region.

With the help of this modal language, semantic arguments about circuit histories can be expressed concisely. We may condense the specification of delay elements into a few syntactic primitives that completely do away with time variables and quantification. Specifically, we may specify that a formula ψ takes on the truth value of another formula ϕ with a delay of at most D, i.e., whenever ϕ becomes true (false) in any history then ψ is true (false) within D-time, too. Such a delay is given by the formula $(\neg\phi \supset \Diamond_D \neg\psi) \wedge (\phi \supset \Diamond_D \psi)$, or equivalently, $(\phi \equiv 0 \supset \Diamond_D \neg\psi \equiv 0) \wedge (\phi \equiv 1 \supset \Diamond_D \psi \equiv 1)$, which we will abbreviate as $\psi :=_D \phi$. Note that $\psi :=_0 \phi$ is equivalent $\phi \equiv \psi$ i.e., both are validated by the same histories.

Lemma 1 *Let Z, z be two network vertices. $h \models z :=_D Z$ iff h behaves so that z and Z are connected through an UN-delay according to Definition 5, i.e., if h satisfies $UN_D(Z,z)$.*

Our language is a *constructive* extension of Boolean algebra, which is reflected in the strong form of negation and its interplay with disjunction. Consider that $h\,I \models \neg\phi$ iff ϕ is false in *all* sub-intervals $J \subseteq I$, which is stronger than just requiring ϕ to be false in I. Consequently, we may well have neither $h\,I \models \phi$ nor $h\,I \models \neg\phi$, namely if ϕ is true in

some sub-intervals of I and false in others. In such a case we have $h\,I \not\models \phi \vee \neg\phi$ which means that the classical Law of the Excluded Middle is not valid for the constructive operators $\{\vee, \neg\}$. The absence of this law is what gives the necessary extra expressiveness over Boolean algebra to handle delays and signal instabilities. For instance, if z is a circuit vertex then $h\,I \models z \vee \neg z$, or equivalently $h\,I \models z \equiv 1 \vee z \equiv 0$, holds exactly if signal z is constant (stable) during interval I in history h. Abbreviating $z \vee \neg z$ by $z{\downarrow}$ we can think of UN-Logic as an extension of Boolean algebra by means to express stability $\downarrow$ and time delay $\Diamond_D$. Specifically, the UN-delay $z :=_D Z$ (Lemma 1) could also be specified by the formula $Z{\downarrow} \supset \Diamond_D(z{\downarrow} \wedge Z \equiv z)$, which says that if the delay input Z is stable then with some (maximum) delay D output z will become stable as well, and it will assume the same value as Z. This formulation motivates our notation $z :=_D Z$ suggesting an assignment. Observe that $\psi :=_0 \phi$ is equivalent to $\psi \equiv \phi$ and $\Diamond_D\phi$ is equivalent to $\phi :=_D 1$. This gives another way of obtaining UN-Logic, viz. as an extension of Boolean algebra by stability $z{\downarrow}$ and a delayed assignment $:=_D$, instead of $\Diamond_D$.

Although interesting in its own right we will not be developing the full theory of UN-Logic and its algebraic properties here (see also Sect. 4). For the purpose of specifying combinational circuit networks all we need is the special fragment consisting of conjunctions of formulas of the shape $S :=_D R$, essentially, for regions or Boolean expressions R, S and $D \geq 0$. Somewhat more generally, we shall give an analysis of the *generalized modal Horn-clause fragment* that consists of all timed regions $\Diamond_D R$ with $D \geq 0$ and *timed clauses* $R \supset \Diamond_D S$, for $D > 0$ with arbitrary R, S. We will call sets Φ of such formulas *generalized modal Horn-clause theories* or simply *theories*. A special class of theories are the *network theories* arising in the specification of circuit networks. Their application will now be explained in the next section.

3.1.1 Network specifications

A network N is described by a set or conjunction of *vertex formulas* of the form $v :=_D V$ with $D > 0$ or $v \equiv V$, where v is a vertex in N and V the *vertex function* corresponding to v. Both kinds of formulas logically connect a vertex with its vertex function. The former, $v :=_D V$, introduces a delay at vertex v, making v a state vertex, and the latter $v \equiv V$ defines v to be a combinational node identified with its vertex function. The conjunction of all vertex formulas, one for each vertex, yields a formula ϕ_N that specifies the real-time executions of N.

In general, a network specification in *substitution normal form (snf)* has the shape

$$\phi_N^{\mathrm{snf}} = \bigwedge_{i=1}^{m} s_i :=_{D_i} S_i \;\wedge\; \bigwedge_{j=1}^{k} v_j \equiv F_j.$$

The expression S_i in each state clause $s_i :=_{D_i} S_i$ is the *excitation function* of s_i and the expression F_j in the equation $v_j \equiv F_j$ for a combinational vertex is the *circuit function* of v_j. The excitation and circuit functions S_i, F_j only depend on input variables $\mathcal{X}$ and state variables $\mathcal{S}$ but not on the combinational vertices $v_j \in \mathcal{Z} \setminus (\mathcal{X} \cup \mathcal{S})$. The substitution normal form always exists, provided N is well-formed.

The advantage of the substitution normal form is that if we are interested only in the behavior of the state signals s_i, then we can simply drop all equations $v_j \equiv F_j$ without losing information. This yields the *reduced snf (rsnf)*

$$\phi_N^{\mathrm{rsnf}} = \bigwedge_{i=1}^{m} s_i :=_{D_i} S_i.$$

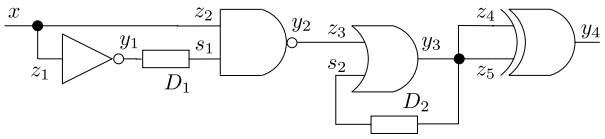


Fig. 10 Gate network N_2

It is not difficult to see that if ψ is a formula that only mentions state and input variables then $\phi_N \models \psi$ if and only if $\phi_N^{\text{rsnf}} \models \psi$. That is, ϕ_N^{rsnf} contains full information regarding state behavior.

Example 6 Consider the gate network N_2 in Fig. 10 in which the input and output ports of all gates are named vertices, $\mathcal{Z} = \{x, y_1, \ldots, y_4, z_1, \ldots, z_5, s_1, s_2\}$. Among those, $\mathcal{X} = \{x\}$ is the only primary input variable and $\mathcal{S} = \{s_1, s_2\} \subseteq \mathcal{Z}$ are the state variables. The specification of this network is

$$\phi_{N_2} =_{df} y_1 \equiv \overline{z_1} \ \wedge\ y_2 \equiv \overline{z_2 \cdot s_1} \ \wedge\ y_3 \equiv z_3 + s_2 \ \wedge\ y_4 \equiv z_4 \cdot \overline{z_5} + \overline{z_4} \cdot z_5 \ \wedge\ z_1 \equiv x$$
$$\wedge\ z_2 \equiv x \ \wedge\ z_3 \equiv y_2 \ \wedge\ z_4 \equiv y_3 \ \wedge\ z_5 \equiv y_3 \ \wedge\ s_1 :=_{D_1} y_1 \ \wedge\ s_2 :=_{D_2} y_3.$$

A formal calculus for dealing with such specifications will be developed below in Sect. 3.2. In particular, it is semantically sound to perform substitution of equals for equals. For instance, since ϕ_{N_2} contains the equation $y_3 \equiv z_3 + s_2$, we can replace every occurrence of y_3 in ϕ_{N_2} by $z_3 + s_2$ without changing the semantics. Thus, $y_3 \equiv z_3 + s_2 \wedge s_2 :=_{D_2} y_3$ is equivalent to $y_3 \equiv z_3 + s_2 \wedge s_2 :=_{D_2} z_3 + s_2$, which effectively eliminates the dependency of s_2 on the combinational vertex y_3. In this way we can reformulate ϕ_{N_2} so that all vertices are specified in terms of the input x and state variables s_1, s_2 alone. Also, we can apply standard Boolean reasoning in this substitution process. For example, since $z_4 \equiv y_3$ and $z_5 \equiv y_3$ we have $y_4 \equiv z_4 \cdot \overline{z_5} + \overline{z_4} \cdot z_5 \equiv y_3 \cdot \overline{y_3} + \overline{y_3} \cdot y_3 \equiv 0 + 0 \equiv 0$. In other words, the equation $y_4 \equiv z_4 \cdot \overline{z_5} + \overline{z_4} \cdot z_5$ in ϕ_N can be simplified to $y_4 \equiv 0$. If we eliminate the combinational vertices $\mathcal{Z} \setminus (\mathcal{X} \cup \mathcal{S}) = \{z_1, \ldots, z_5, y_1, \ldots, y_4\}$ in this fashion from the right-hand sides of identities and assignments, we get:

$$\phi_{N_2} \equiv y_1 \equiv \overline{x} \ \wedge\ y_2 \equiv \overline{x \cdot s_1} \ \wedge\ y_3 \equiv \overline{x \cdot s_1} + s_2 \ \wedge\ y_4 \equiv 0$$
$$\wedge\, z_1 \equiv x \ \wedge\ z_2 \equiv x \ \wedge\ z_3 \equiv \overline{x \cdot s_1} \ \wedge\ z_4 \equiv \overline{x \cdot s_1} + s_2$$
$$z_5 \equiv \overline{x \cdot s_1} + s_2 \wedge s_1 :=_{D_1} \overline{x} \ \wedge\ s_2 :=_{D_2} \overline{x \cdot s_1} + s_2.$$

At this point, the description of each vertex is self-contained in terms of state and input nodes. This is a substitution normal form (snf) that is uniquely defined (up to equivalences of Boolean expressions) by the condition that no further substitutions are possible. In the example, $\overline{x}$ and $\overline{x s_1} + s_2$ are the excitation functions of states s_1 and s_2 respectively, and $\overline{x s_1} + s_2$ is the circuit function of nodes y_3, z_4, z_5. The reduced snf obtained by ignoring the combinational nodes $\mathcal{Z} \setminus (\mathcal{X} \cup \mathcal{S}) = \{y_1, \ldots, y_4, z_1, \ldots, z_5\}$, gives

$$\phi_{N_2}^{\text{rsnf}} \ \equiv\ s_1 :=_{D_1} \overline{x} \ \wedge\ s_2 :=_{D_2} \overline{x s_1} + s_2$$

as the specification of the system.

In constructing a snf/rsnf it is important to keep in mind that a state excitation $s :=_D S$ is not an equation but an abbreviation for $S \equiv 0 \supset \Diamond_D(s \equiv 0) \wedge S \equiv 1 \supset \Diamond_D(s \equiv 1)$. In

particular, we cannot use $s :=_D S$ to substitute occurrences of s in ϕ_N by the right-hand side S. Consider a network theory such as $y :=_1 \overline{x} \wedge z :=_1 xy$. If these were mistaken for equations we might be tempted to substitute $\overline{x}$ for y in the right-hand side of $z :=_1 x \cdot y$ to infer $z :=_1 x \cdot \overline{x}$, which is the same as $z :=_1 0$, since $x\overline{x} \equiv 0$. However, the circuits $y :=_1 \overline{x} \wedge z :=_1 x \cdot y$ and $y :=_1 \overline{x} \wedge z :=_1 0$ have different behavior: in the latter z is constant 0 within 1 time unit, independent of input x, while in the former z is constant only if x is guaranteed to be constant, too, otherwise z may exhibit glitches.

The formula ϕ_N constructed above, regardless whether snf/rsnf or not, specifies the UN-histories of N under all possible input stimuli. For convenience, we will break up all conjunctions in ϕ_N and handle ϕ_N directly as a set or *network theory* Φ_N containing vertex formulas $v_j \equiv V_j$ (viewed as regions) and $s_i :=_{D_i} S_i$ (a pair of implications). We will also want to provide specific input signals, using the following abbreviations: Given input vertices $\mathcal{X} = \{x_1, \ldots, x_n\}$ and a vector $a = \langle a_1, \ldots, a_n \rangle$ of input values $a_i \in \mathbb{B}$, we write $\mathcal{X} \equiv a$ for the region $\bigwedge_{i=1}^{n} x_i \equiv a_i$. Then, we use the notation $\Phi_{N,a}$ for the combination $\Phi_N \cup \{\mathcal{X} \equiv a\}$ which adds the constant input a to the specification of N.

Theorem 1 (Semantical adequacy) *Let N be a network, a some input vector and $\phi_{N,a}$ the associated network theory.*

(i) *For all histories h, $h \in UN$-history(N, a) iff $h \models \Phi_{N,a}$.*
(ii) UN-transient(N, a) *is the largest $R \subseteq \mathbb{B}^S$ such that $\Phi_{N,a} \models \Diamond_D \overline{R}$ for some $D \geq 0$.*
(iii) UN-recurrent(N, a) *is the smallest $R \subseteq \mathbb{B}^S$ with $\Phi_{N,a} \models \Diamond_D R$ for some $D \geq 0$.*

Theorem 1 is a direct consequence of Lemma 1. It means that the logical syntax of UN-Logic is sufficiently expressive to handle steady-state behavior of a network. The result tells us that deciding the sets of transient and recurrent states is equivalent to deciding the modal theory of N, i.e., the relation $\Phi_{N,a} \models \Diamond_D R$. In particular, we obtain the following important corollary:

Corollary 1 (Logical characterization of constructive networks) *A network N is constructive iff for every input a there is a unique state s such that $\Phi_{N,a} \models \Diamond_D \{s\}$ for some $D \geq 0$ and there does not exist any $D \geq 0$ with $\Phi_{N,a} \models \Diamond_D \emptyset$.*

Proof The corollary can be shown as follows: Fixing N and a, constructivity means by Definition 1 that UN-recurrent$(N, a) = \{s\}$ for a unique state s. Because of Theorem 1(iii) this is the same as saying that $\{s\}$ is the smallest region such that $\Phi_{N,a} \models \Diamond_D \{s\}$ for any D. This, in turn, is equivalent to the two conditions (i) $\Phi_{N,a} \models \Diamond_D \{s\}$ for some D and (ii) whenever $\Phi_{N,a} \models \Diamond_{D'} R$ then $s \in R$. Now, the latter can be reformulated to the condition (ii') that there is no $D' \geq 0$ with $\Phi_{N,a} \models \Diamond_{D'} \emptyset$. The direction (ii) $\Rightarrow$ (ii′) is obvious and the converse $\neg$(ii) $\Rightarrow \neg$(ii′) holds in the presence of (i) because if $\Phi_{N,a} \models \Diamond_{D'} R$ such that $s \notin R$ and also $\Phi_{N,a} \models \Diamond_D \{s\}$ we have $\Phi_{N,a} \models \Diamond_{max(D,D')}(\{s\} \cap R)$ by the properties of UN-Logic, which means $\Phi_{N,a} \models \Diamond_{max(D,D')} \emptyset$ considering that $\{s\} \cap R = \emptyset$. □

What remains to be seen, however, is how we can decide $\models$. If we tried to do that directly in terms of histories nothing is gained by introducing UN-Logic. Instead, we study the proof-theoretic properties of the $\models$ relation. Following the standard approach in logics, we replace $\models$ by a corresponding notion $\vdash$ of deductive entailment and set up a formal calculus to derive facts of the form $\Phi \vdash \Diamond_D R$. This will be done in the next section Sect. 3.2. Once we have shown that the calculus is sound and complete, we know $\vdash$ maintains enough

$$\Diamond true: \frac{}{\Diamond_E 1} \qquad \supset\Diamond L: \frac{\Diamond_D R}{\Diamond_{D+E} S}\ R \supset \Diamond_E S \in \Phi$$

$$\Diamond id: \frac{\Diamond_D R}{\Diamond_E S}\ D \leq E, R \subseteq S \qquad \Diamond\wedge R: \frac{\Diamond_D S \quad \Diamond_E T}{\Diamond_{max(D,E)} S \cap T}$$

Fig. 11 A simulation calculus for UN-delay network theories

information to handle our real-time models symbolically. We will then, in a second step, reconstruct from the deduction system a new, this time more economic, semantic domain in which the calculus can be adequately interpreted. This will be the ternary signal domain—re-discovered and extended with quantitative timing parameters—in Sect. 3.3.

3.2 UN-calculus: the symbolic simulation of UN-delay networks

Our simulation calculus consists of a formal derivation relation of the form $\Phi \vdash \Theta$, called a *sequent*, where Φ is a theory and Θ a non-empty set of timed regions. The elements of Φ are referred to as the *hypotheses* and those of Θ as the *conclusions*. The theory Φ plays the role of a *system specification*, given by the *conjunction* of all its formulas, and Θ is a *stabilization bound* obtained as the *disjunction* of all its timed regions. It is technically convenient for the completeness proof to permit both sets to be infinite, subject to the *non-Zeno-ness* condition (see Definition 9 below) that they not contain an infinitely dense collection of delay values.

Our calculus to derive sequents consists of the four derivation rules $\Diamond id$, $\Diamond true$, $\Diamond\wedge R$ and $\supset\Diamond L$ shown in Fig. 11. They allow us to draw formal conclusions from a set of hypotheses according to the following definition:

Definition 8 (Derivability) We say a timed region $\Diamond_D R$ is *derivable* from Φ, i.e., $\Phi \vdash \Diamond_D R$, if there exists a finite sequence of timed regions

$$\Diamond_{D_0} R_0,\ \Diamond_{D_1} R_1,\ \Diamond_{D_2} R_2,\ \ldots,\ \Diamond_{D_n} R_n$$

such that $D_n = D$, $R_n = R$ and for every $0 \leq j \leq n$ the timed region $\Diamond_j R_j$ is contained in Φ or obtained by applying of one of the rules from Fig. 11 to the timed regions $\{\Diamond_i R_i \mid i < j\}$ appearing earlier in the list. In particular, this means that $\Diamond_{D_0} R_0 \in \Phi$. We write $\Phi \vdash \Theta$ if there exists some $\theta \in \Theta$ such that $\Phi \vdash \theta$.

The semantical meaning of $\Phi \vdash \Theta$ is the statement that if a history h (in some interval) satisfies all hypotheses in Φ then it must necessarily satisfy at least one of the conclusions in Θ. Formally, we stipulate that a sequent $\Phi \vdash \Theta$ is *valid*, written $\Phi \models \Theta$, iff for all histories h and intervals $[s,t)$ it is the case that

$$\big(\forall \phi \in \Phi.\ h[s,t) \models \phi\big) \Rightarrow \exists \theta \in \Theta.\ h[s,t) \models \theta.$$

Note that if $\Phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$ and $\Theta = \{\theta_1, \theta_2, \ldots, \theta_m\}$ are finite then $\Phi \models \Theta$ is the same as $\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n \models \theta_1 \vee \theta_2 \vee \cdots \vee \theta_m$. For singleton sets $\Theta = \{\theta\}$, $\Phi \models \Theta$ coincides with $\Phi \models \theta$ as defined above in Sect. 3.1. Verifying soundness of the calculus is straightforward by induction on the structure of derivations:

Theorem 2 (Soundness) $\Phi \vdash \Theta \Rightarrow \Phi \models \Theta$.

Applied to a network theory $\Phi_{N,a}$ the rules of the calculus essentially amount to a symbolic simulation of N under input a. To understand how the calculus works and why it is sound let us look at the rules one by one:

- Axiom $\Diamond$*true* says that timed regions of the form $\Diamond_E 1$ are a stabilization bound for any theory Φ. Soundness of this is obvious considering that 1 represents the full total space $\mathbb{B}^Z$ and thus $h[s,t) \models \Diamond_E 1$ whatever $h[s,t)$ and E. This axiom is not important for symbolic simulation but necessary for reasons of completeness.
- The rule $\supset\Diamond L$ implements *vertex switching*: Suppose our theory Φ contains the implication $R \supset \Diamond_D S$, which says that whenever (during some interval) a history moves inside region R, it converges into S with a delay of D. Then, if we can derive the bound $\Diamond_E R$ from Φ we know that all histories of Φ are guaranteed to stabilize in region R within E time. Thus, adding up both delays $D + E$ yields a lower bound on the time that any history of Φ must enter region S, i.e., $\Diamond_{D+E} S$. Recall that network theories $\Phi_{N,a}$ (see Sect. 3.1.1) consist of vertex functions $v :=_D V$ which split into two implications $V \supset \Diamond_D v$ and $\overline{V} \supset \Diamond_D \overline{v}$. The rule $\supset\Diamond L$ then permits us to pass from $\Diamond_E V$ to $\Diamond_{D+E} v$ and from $\Diamond_D \overline{V}$ to $\Diamond_{D+E} \overline{v}$. This amounts to switching vertex v on or off, respectively.
- Rule $\Diamond id$ amounts to *Boolean function evaluation*. Formally, it is a weakening rule which permits us to increase the stabilization time and region in any derived bound $\Diamond_D R$ to obtain $\Diamond_E S$ where $D \leq E$ and $R \subseteq S$. This is obviously sound since under these conditions, i.e., allowing for more time and a larger region, $h[s,t) \models \Diamond_D R$ implies $h[s,t) \models \Diamond_E S$. In the symbolic simulation of a network $\Phi_{N,a}$, this rule is used to show that under the system's stabilization $\Diamond_D R$ achieved by time D some vertex function V evaluates to true, i.e., that we have $\Diamond_D V$. This requires checking that $R \subseteq V$ when V is seen as a region, or that the assumption $R \equiv 1$ implies $V \equiv 1$ taking R as a Boolean expression.
- Finally, rule $\Diamond\wedge R$ simulates a *concurrent superposition* of computations: Suppose we can derive two stabilization goals $\Diamond_D S$ and $\Diamond_E T$ from Φ. This means that all histories specified by Φ must stabilize in region S within D time and in T within E time. Hence, all histories are guaranteed to be inside the intersection $S \cap T$ if we wait $max(D, E)$ time units, whence $\Diamond_{D+E} S \cap T$ is a valid stabilization bound. The rule $\Diamond\wedge R$ implements fan-in (information re-convergence) in symbolic simulation.

Example 7 For illustration take the network N_2 from Fig. 10 with rsnf specification

$$\phi_{N_2}^{\text{rsnf}} = s_1 :=_{D_1} \overline{x} \;\wedge\; s_2 :=_{D_2} \overline{x \cdot s_1} + s_2,$$

which corresponds to the modal Horn theory

$$\Phi_{N_2} = \{\overline{x} \supset \Diamond_{D_1} s_1,\ \overline{\overline{x}} \supset \Diamond_{D_1} \overline{s_1},\ \overline{x \cdot s_1} + s_2 \supset \Diamond_{D_2} s_2,\ \overline{\overline{x \cdot s_1} + s_2} \supset \Diamond_{D_2} \overline{s_2}\}.$$

Let us consider the behavior of N_2 under the constant input $x \equiv 0$. Looking at the network diagram in Fig. 10 we can see that $x \equiv 0$ propagates through the invertor and forces state vertex s_1 to become 1 after a maximum delay of D_1. Concurrently with this, the input $x \equiv 0$ drives the output y_2 of the NAND-gate high instantaneously, which also holds output y_3 of the OR-gate high without delay. Because of the delay element between y_3 and s_2 it may take up to D_2 time to bring vertex s_2 from its unknown initial state up to 1. Thus, after $max(D_1, D_2)$ delay, both state vertices s_1, s_2 are guaranteed to be stable 1.

How do we derive the stabilization bounds $\Diamond_{D_1}(s_1 \equiv 1)$, $\Diamond_{D_2}(s_2 \equiv 1)$ and $\Diamond_{max(D_1,D_2)}(s_1 \equiv 1 \wedge s_2 \equiv 1)$ from the network theory Φ_{N_2} under $x \equiv 0$, i.e., from $\Phi_{N_2,a} =_{df} \Phi_{N_2} \cup \{\Diamond_0 \overline{x}\}$, formally? First, trivially, we have $\Phi_{N_2,a} \vdash \Diamond_0 \overline{x}$ because $\Diamond_0 \overline{x}$ is included in $\Phi_{N_2,a}$. Also, $\Phi_{N_2,a}$

$$\dfrac{\dfrac{\Diamond_0\overline{x}}{\Diamond_{D_1}s_1}\supset\Diamond L \quad \dfrac{\dfrac{\Diamond_0\overline{x}}{\Diamond_0(\overline{x\cdot s_1}+s_2)}\Diamond id}{\Diamond_{D_2}s_2}\supset\Diamond L}{\Diamond_{max(D_1,D_2)}s_1\cdot s_2}\Diamond\wedge R \qquad \dfrac{\dfrac{\Diamond_0 x}{\Diamond_{D_1}\overline{s_1}}\supset\Diamond L \quad \dfrac{\dfrac{\Diamond_0 x}{\Diamond_{D_1}\overline{s_1}}\supset\Diamond L:\overline{\overline{x}}\equiv x}{\Diamond_{D_1}(\overline{x\cdot s_1}+s_2)}\Diamond id}{\Diamond_{D_1+D_2\overline{s_1}}\cdot s_2}\Diamond\wedge R$$

Fig. 12 Symbolic simulation of gate network N_2 under $x\equiv 0$ (*left*) and $x\equiv 1$ (*right*)

contains the vertex formula $\overline{x}\supset\Diamond_{D_1}s_1$ so that we derive $\Phi_{N_2,a}\vdash\Diamond_{D_1}s_1$ by rule $\supset\Diamond L$, considering $D_1+0=D_1$. We can also use $\Phi_{N_2,a}\vdash\Diamond_0\overline{x}$ to evaluate the vertex function $\overline{x\cdot s_1}+s_2$ in the vertex formula $\overline{x\cdot s_1}+s_2\supset\Diamond_{D_2}s_2$ of s_2 as follows: As a Boolean expression $\overline{x}$ is equivalent to $\overline{x}\equiv 1$, so that $\overline{x\cdot s_1}+s_2\equiv\overline{x}+\overline{s_1}+s_2\equiv 1+\overline{s_1}+s_2\equiv 1$. This shows that $\overline{x}\equiv 1$ implies $\overline{x\cdot s_1}+s_2\equiv 1$ or $\overline{x}\subseteq\overline{x\cdot s_1}+s_2$ reading the expressions as regions. Hence, we have $\Phi_{N_2,a}\vdash\Diamond_0(\overline{x\cdot s_1}+s_2)$ by rule $\Diamond id$. Now we can invoke the implication $\overline{x\cdot s_1}+s_2\supset\Diamond_{D_2}s_2$ from $\Phi_{N_2,a}$ and derive $\Phi_{N_2,a}\vdash\Diamond_{D_2}s_2$ by rule $\supset\Diamond L$. Finally, we combine $\Phi_{N_2,a}\vdash\Diamond_{D_1}s_1$ and $\Phi_{N_2,a}\vdash\Diamond_{D_2}s_2$ by rule $\Diamond\wedge R$ to obtain $\Phi_{N_2,a}\vdash\Diamond_{max(D_1,D_2)}(s_1\cap s_2)$, which is precisely $\Phi_{N_2,a}\vdash\Diamond_{max(D_1,D_2)}(s_1\equiv 1\wedge s_2\equiv 1)$ because $s_1\cap s_2$ and $s_1\equiv 1\wedge s_2\equiv 1$ denote the same region, also represented by the Boolean expression $s_1\cdot s_2$. The formal derivation corresponding to this simulation is seen in Fig. 12 on the left.

Now take N_2 under input $x\equiv 1$, i.e., $\Phi_{N_2,a}=_{df}\Phi_{N_2}\cup\{\Diamond_0 x\}$. Again, the network stabilizes at $s_2\equiv 1$ but this time it is the second input s_1 that controls the NAND-gate's output and brings it up as $y_2\equiv 1$. Therefore, we first have to wait for s_1 to stabilize at 0, which yields an overall delay of D_1+D_2 for s_2. Hence, we get $\Diamond_{D_1+D_2}(s_1\equiv 0\wedge s_2\equiv 1)$ rather than $\Diamond_{D_2}s_2$ as before. The symbolic simulation runs like this: Our initialization justifies $\Phi_{N_2,a}\vdash\Diamond_0 x$. Since $x\equiv\overline{\overline{x}}$, we can activate the vertex formula $\overline{\overline{x}}\supset\Diamond_{D_1}\overline{s_1}$ in $\Phi_{N_2,a}$ using rule $\supset\Diamond L$ to obtain $\Phi_{N_2,a}\vdash\Diamond_{D_1}\overline{s_1}$. Assuming $\overline{s_1}$, or equivalently $s_1\equiv 0$, we evaluate $\overline{x\cdot s_1}+s_2\equiv\overline{x\cdot 0}+s_2\equiv\overline{0}+s_2\equiv 1+s_2\equiv 1$. This proves the inclusion $\overline{s_1}\subseteq\overline{x\cdot s_1}+s_2$ of regions which yields $\Phi_{N_2,a}\vdash\Diamond_{D_1}\overline{x\cdot s_1}+s_2$ by rule $\Diamond id$. At this point we are entitled to apply rule $\supset\Diamond L$ on the vertex function $\overline{x\cdot s_1}+s_2\supset\Diamond_{D_2}s_2$ from $\Phi_{N_2,a}$ concluding $\Phi_{N_2,a}\vdash\Diamond_{D_1+D_2}s_2$ as desired. Finally, together with $\Phi_{N_2,a}\vdash\Diamond_{D_1}\overline{s_1}$ an application of $\Diamond\wedge R$ derives $\Phi_{N_2,a}\vdash\Diamond_{D_1+D_2\overline{s_1}}\cdot s_2$ since $max(D_1,D_1+D_2)=D_1+D_2$ and $(\overline{s_1}\cap s_2)\equiv\overline{s_1}\cdot s_2$. The formal derivation of this is seen in Fig. 12 on the right.

Ignoring the modal operator and the fact that regions are not really atomic but sets of Boolean vectors, the calculus is simply a propositional version of Prolog. The rules $\Diamond\wedge R$ and $\supset\Diamond L$ are essentially the rules known as the *decomposition* and *backchaining* rule, respectively, of the Prolog SLD-resolution mechanism [22]. The rule $\Diamond id$ is a special case of backchaining for atomic clauses. Pushing this analogy, the relation $\Phi\vdash\Diamond_D S$ could be seen as a modalized version of the SLD-resolution relation with program Φ and answer goal $\Diamond_D S$. One could mechanize the calculus in Constraint Prolog (with Boolean and arithmetic constraints) to obtain an executable implementation. The implementation of the calculus is not our primary concern, though. For our purposes, it acts as an interface between the logic and algebraic viewpoints and a reference system for reducing the correctness and exactness of Malik's ternary simulation algorithm to soundness and completeness of a particular logic theory.

Suppose Φ_N is the specification of network N and we wish to constrain N to stay within a predefined region $R\subseteq\mathbb{B}^Z$ at all times. Typically, R might restrict the input signals to have certain constant values, but it could also specify some other region of the state space.

Exploring the logic consequences of $\Phi_N, \Diamond_0 R$ under $\vdash$ amounts to a formal analysis of N's steady-state behavior within region R under the UN-delay model, by virtue of the Soundness Theorem 2. For instance, if $\Phi_N, \Diamond_0 R \vdash \Diamond_D S$ we know that assuming N stays in region R for long enough,[5] it will necessarily enter region S after at most D time. If, however, $\Phi_N, \Diamond_0 R \vdash \Diamond_D 0$, then (since N cannot enter the empty region) we know N will leave R before D time. In other words, R is a region with maximal life time D. By varying the triple (R, D, S) such that $\Phi_N, \Diamond_0 R \vdash \Diamond_D S$ we can gather *correct* information about the transient and steady-state behavior of N.

We will show below in Theorem 3 that our calculus is complete, in particular that if $\Phi_N, \Diamond_0 R \models \Diamond_D S$ then $\Phi_N, \Diamond_0 R \vdash \Diamond_D S$. If each waveform of N that stays inside R must enter S within D time, then $\Diamond_D S$ is formally derivable from the assumption $\Diamond_0 R$. More concretely, suppose some signal s_i must stabilize at α in bounded time provided input $\mathcal{X}$ is kept constant at a. Then $\Phi_N, \mathcal{X} \equiv a \models \Diamond_D(s_i \equiv \alpha)$ must hold for some D. Completeness guarantees that the calculus will actually derive this fact, i.e., $\Phi_N, \mathcal{X} \equiv a \vdash \Diamond_D(s_i \equiv \alpha)$. Or, put negatively, if we cannot derive $\Phi_N, \mathcal{X} \equiv a \vdash \Diamond_D(s_i \equiv \alpha)$ then N has a history which after time D still has $s_i \neq \alpha$ under input $\mathcal{X} \equiv a$. As another example suppose we find that $\Phi_N, \Diamond_0 R \nvdash \Diamond_D 0$ for any D. Then, by completeness, we conclude N can stay inside R for an arbitrarily long time. Hence, at least one state in R must be recurrent. If we know R is minimal, then all states in R are recurrent. In this way, completeness guarantees that analyzing N in terms of $\vdash$ yields *exact* information about the steady-state behavior of N.

We are now ready to state the Completeness Theorem, which, like the Soundness Theorem 2 is done for arbitrary generalized modal Horn clause theories Φ, not only those that arise from circuit networks, assuming the following non-Zeno-ness condition:

Definition 9 Let Ξ be a set of generalized modal Horn clauses (hypotheses Φ or conclusions Θ of a sequent). Let $dvals(\Xi) \subseteq \mathbb{R}^+$ be the set of delay values occurring in Ξ, i.e.,

$$dvals(\Xi) =_{df} \left\{E \mid \exists R \subseteq \mathbb{B}^{\mathcal{Z}}.\ \Diamond_E R \in \Xi\right\} \cup \left\{E \mid \exists R, T \subseteq \mathbb{B}^{\mathcal{Z}}.\ R \supset \Diamond_E T \in \Xi\right\}.$$

A sequent $\Phi \vdash \Theta$ is called *non-Zeno* if no bounded interval $[s, t) \subset \mathbb{R}^+$ contains an infinite *decreasing* sequence $a_0 > a_1 > a_2 > \cdots$ of delay values $a_i \in dvals(\Phi) \cap [s, t)$ from the hypotheses Φ or an infinite *increasing* sequence $b_0 < b_1 < b_2 < \cdots$ of delay values $b_i \in dvals(\Theta) \cap [s, t)$ from the conclusions Θ. We say the formula sets Φ, Θ are *non-Zeno* in case the respective condition holds.

Theorem 3 (Completeness) *Let $\Phi \nvdash \Theta$ be an unprovable non-Zeno sequent. Then, $\Phi \not\models \Theta$.*

Proof The proof of the Completeness Theorem 3 applies a standard technique of constructing syntactic counter models for logic theories. Starting from the assumption $\Phi \nvdash \Theta$ one constructs a single history h, called a *counter model*, which validates Φ and falsifies Θ. Formally, $h[0, \infty) \models \phi$ and $h[0, \infty) \not\models \theta$ for all $\phi \in \Phi$ and $\theta \in \Theta$. This proves $\Phi \not\models \Theta$. The construction of h from Φ and Θ uses a syntactic saturation process, in which the set Φ is maximally extended to a theory Φ^*, consistently with the falsification constraints Θ, so that Φ^* contains enough information to specify h uniquely. Concretely, Φ^* describes the waveform of h explicitly in terms of formulas $\Diamond_\tau s_i \equiv b$ meaning that $h_i(\tau) = b$. The details, which are somewhat involved, do not contribute much to our story here. They can be found in [30].

[5] This is always possible if R only constrains the input.

Let us indicate why we need the non-Zeno-ness condition. Consider a sequent $\Phi \vdash \Diamond_0(s \equiv 0)$ with $\Phi = \{\Diamond_{1/n+1}(s \equiv 0) \mid n \in \mathbb{N}\}$. Obviously, Φ is Zeno because $dvals(\Phi) = \{1/(n+1) \mid n \in \mathbb{N}\}$ contains an infinite descending chain of delay indices in the interval $[0, 1)$. It is not difficult to see that the sequent is semantically valid, i.e., $\Phi \models \Diamond_0(s \equiv 0)$. Any history satisfying all of $\Diamond_{1/n+1}(s \equiv 0)$ for $n \in \mathbb{N}$ simultaneously on $[0, 1)$ must have signal $s \equiv 0$ be at 0 arbitrarily close to 0. But this means it must be constant $s \equiv 0$ throughout $[0, 1)$, whence $h[0, 1) \models \Diamond_0(s \equiv 0)$. On the other hand, the sequent $\Phi \vdash \Diamond_0(s \equiv 0)$ cannot be provable in the calculus for if it were, we would have $\Phi_{fin} \vdash \Diamond_0(s \equiv 0)$ for a *finite* subset $\Phi_{fin} \subset \Phi$ as the proof can only depend on a finite number of hypotheses. But then the conclusion $\Diamond_0(s \equiv 0)$ is not valid any more because Φ_{fin} no longer constrains the history to be 0 *infinitely* close to 0. By soundness, since $\Phi_{fin} \not\models \Diamond_0(s \equiv 0)$, we must have $\Phi_{fin} \not\vdash \Diamond_0(s \equiv 0)$ contradicting completeness. Analogously, one can show that non-Zeno-ness is necessary on the conclusion side Θ of a sequent. □

The key benefit of the Completeness Theorem is that it allows us to reduce questions about the UN-delay behavior of networks to questions about the syntactic and algorithmic properties of a logic calculus. We will exploit this below in Sect. 3.3 where we identify ternary simulation as a complete proof procedure to handle network theories, and thus—by way of the Completeness Theorem—UNstabilization behavior. Another important property of UN network theories which can be derived from the calculus is constructivity:

Theorem 4 (Constructivity) *Let Φ be a theory and Θ a non-empty set of modalized regions, which are non-Zeno in the sense of Definition 9. Then, $\Phi \models \Theta$ implies there exists a single $\theta \in \Theta$ such that $\Phi \models \theta$.*

Proof Suppose we have $\Phi \models \Theta$. Then, because of the Completeness Theorem 3 the sequent $\Phi \vdash \Theta$ is derivable in the simulation calculus. But this means, by definition, there already exists a single $\theta \in \Theta$ with $\Phi \vdash \theta$. From the Soundness Theorem 2 we then conclude $\Phi \models \theta$ as claimed. □

Theorem 4 generalizes the standard notion of constructivity for logic theories, known as the *disjunction property*: If a disjunction is universally valid in the theory Φ then one of the disjuncts is valid. Formally, if $\Phi \models \phi \vee \psi$ then $\Phi \models \phi$ or $\Phi \models \psi$. This is not true of Boolean logic in which the Law of the Excluded Middle always forces $\models \phi \vee \neg\phi$ even in cases where neither $\models \phi$ nor $\models \neg\phi$ holds. Constructivity has the following remarkable consequence for the UN-behavior of networks:

Corollary 2 *Let N be a UN-network under constant input $\mathcal{X} \equiv a$ for some $a \in \mathbb{B}^n$. Then, every state signal either oscillates or stabilizes at a unique value in bounded time. More precisely, for every $s_i \in \mathcal{S}$ exactly one of the following two cases applies*:

Bounded Stability: $\exists \alpha \in \mathbb{B}.\, \exists \tau \geq 0,\ \forall h \in UN\text{-history}(N, a).\, \forall t \geq \tau.\, h_i(t) = \alpha.$
Oscillation: $\exists h \in UN\text{-history}(N, a).\, \forall \alpha \in \mathbb{B}.\, \forall \tau \geq 0.\, \exists t \geq \tau.\, h_i(t) \neq \alpha.$

Proof The proof of Corollary 2 is surprisingly straightforward once we have established constructivity of network theories. Let N be a UN-network and a some input. Consider a state signal $s_i \in \mathcal{S}$ and the set $\Theta = \{\Diamond_\tau(s_i \equiv \alpha) \mid \tau \in \mathbb{N}, \alpha \in \mathbb{B}\}$ of timed regions, which, taken as an infinite disjunction, specifies that s_i stabilizes to some value at some time. Obviously, Θ is non-Zeno (no infinite increasing sequence of time parameters in any bounded

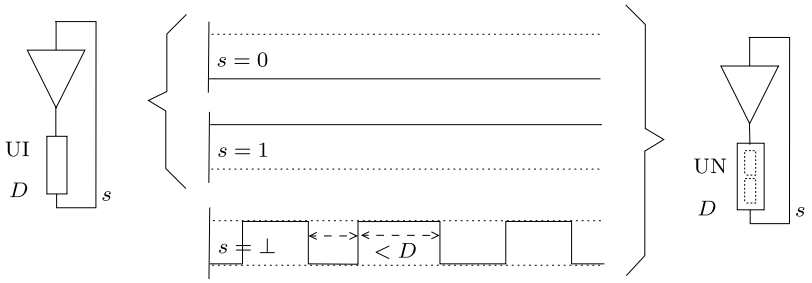


Fig. 13 Delay loop under UI-delay and UN-delay assumptions

interval) since we use the natural numbers as delay indices. Now suppose that indeed $\Phi_{N,a} \models \Theta$, where $\Phi_{N,a} = \Phi_N \cup \{\mathcal{X} \equiv a\}$ specifies UN-history(N, a) (Theorem 1(i)). By the Constructiveness Theorem 4 there exists $\tau \in \mathbb{N}$ and $\alpha \in \mathbb{B}$ such that $\Psi_{N,a} \models \Diamond_\tau(s_i \equiv \alpha)$. This implies that every $h \in$ UN-history(N, a) satisfies $h_i(t) = \alpha$ for all $t \geq \tau$. This yields the first case of Corollary 2. Now, consider what it means that $\Phi_{N,a} \not\models \Theta$. By definition this implies there is a history $h \in$ UN-history(N, a) such that $h \not\models \Diamond_\tau(s_i \equiv \alpha)$ for any $\tau \in \mathbb{N}$ and $\alpha \in \mathbb{B}$. But this is precisely the statement that signal s_i oscillates in h, thus establishing the second case of Corollary 2. □

Example 8 A simple example to illustrate the difference between the UN-delay model which does and the UI-delay model which does not satisfy constructivity is the delay loop, as depicted in Fig. 13. Due to its inertiality, the UI-delay loop behaves like a perfect binary memory element which has exactly two behaviors, $s \equiv 0$ or $s \equiv 1$, seen on the left of the figure. No switching is possible and the initial state of s at time $t = 0$ remains constant forever. Thus, the UI-delay loop is non-deterministic without oscillation, violating Corollary 2. Also, it holds that $\models s \equiv 0 \vee s \equiv 1$ since every history h has $h[0, \infty) \models s \equiv 0$ or $h[0, \infty) \models s \equiv 1$ *but* we have neither $\models s \equiv 0$ nor $\models s \equiv 1$ since none is true of all histories. Depending on the initial state either $h[0, \infty) \not\models s \equiv 0$ or $h[0, \infty) \not\models s \equiv 1$. This violates the Constructiveness Theorem 4. Now, contrast this with the UN-delay behavior seen in Fig. 13 on the right. The UN-delay, too, has both constant waveforms $s \equiv 0$ and $s \equiv 1$ among its possible histories. However, it also permits oscillation bounded by D, such as would be possible if the delay consisted of two or more internal UI-delays starting off in an unstable state. The oscillation makes the disjunctive statement $\models s \equiv 0 \vee s \equiv 1$ invalid for this system and so saves Theorem 4 and Corollary 2.

Corollary 2 highlights a deep invariant of constructivity for the UN-model which is intrinsically coherent so that boundedness of reaction and determinism are equivalent. In particular, UN-delay networks cannot implement non-deterministic choice, and thus fair arbitration, in bounded time. This is in line with theoretical investigations initiated by Marino [24] and ample experimental work reported by Kinniment [17] on the metastability of physical systems. Lamport [20] even conjectures that (bounded) non-determinism and (bounded) arbitration are equivalent. This puts the non-constructive UI-delay outside of physical realizability: As shown by the delay loop on the left of Fig. 13, UI-delay networks can produce bounded time nondeterminism. Indeed, a bounded time arbiter can be built from the ideal UI-delay [30] [Sect. 2.3.2, Fig. 9]. Being more conservative, the UN-model is also more realistic than the UI-model. Corollary 2 makes precise Lamport's suggestion [20] that non-

determinism in distributed systems is intimately linked with metastability: Non-determinism without metastability is non-constructive.

Constructivity resolves the paradoxical nature of $\bot$ representing both a known undefined and an unknown defined value. The steady-state behavior of every state signal s_i (and thus of every signal) in a UN-delay network N may be uniquely described by a function $[\![N]\!]_i$, which for every input vector $a \in \mathbb{B}^n$ computes a well-defined stationary response $[\![N]\!]_i(a) \in (\mathbb{B} \times \mathbb{R}^+)_\bot$. If $[\![N]\!]_i(a) = (\alpha, \tau)$ then s_i stabilizes to value α in τ time (in all histories) and τ is the minimal such stabilization time. Otherwise, if $[\![N]\!]_i(a) = \bot$, then s_i oscillates in at least one history. The value set $\mathbb{TS} =_{df} (\mathbb{B} \times \mathbb{R}^+)_\bot$ is the domain of *timed ternary values*. In the next section we show that timed ternary simulation is a decision procedure for $\vdash$ and thus an algorithm for computing the *timed ternary response* function $[\![N]\!] : \mathbb{B}^n \to \mathbb{TS}^m$.

3.3 Timed ternary simulation

There are well-known efficient decision procedures available for Horn theories which can be adapted to generalized modal Horn clauses. In Sect. 3.3.1 we specialize the standard inductive construction to network specifications $\Phi_{N,a}$ which are a particular form of modal Horn theories. As it will transpire, this essentially gives us Malik's linear simulation algorithm over an extended domain of *timed ternary values* which not only covers the functional steady-state behavior but also quantitative timing information. By establishing a close connection between timed ternary vectors and logical deduction in network theories we will finally obtain correctness and exactness of ternary simulation for UN-delays in Sect. 3.3.2.

Suppose we wish to analyze a network N under a fixed constant input. Let us assume, for simplicity, we are only interested in the dynamics of the state vertices, i.e., $\mathcal{Z} = \mathcal{X} \cup \mathcal{S}$ with $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{S} = \{s_1, \ldots, s_m\}$. Thus, the input, state, and total state spaces are $\mathbb{B}^{\mathcal{X}} = \mathbb{B}^n$, $\mathbb{B}^{\mathcal{S}} = \mathbb{B}^m$, $\mathbb{B}^{\mathcal{Z}} = \mathbb{B}^{m+n}$, respectively. Then, the behavior of N will be specified by a theory $\Phi_{N,a} =_{df} \Phi_N, \Diamond_0(\mathcal{X} \equiv a)$, where $\mathcal{X} \equiv a$, as before, abbreviates the region $\bigwedge_{j=1}^n x_j \equiv a_j$ for input vector $a = \langle a_1, a_2, \ldots, a_n \rangle$. The key observation now is that (i) all modalized regions in $\Phi_{N,a}$ are *cubes*, i.e., conjunctions of literals, (ii) derivations from $\Phi_{N,a}$ only ever need to involve regions specified by cubes and that (iii) cubical regions can be represented exactly as ternary vectors.

3.3.1 Timed ternary vectors

Before we go into details, some general remarks are in order about the abstraction involved in the ternary model. Recall that for any $R \subseteq \mathbb{B}^l \subset \mathbb{S}^l$, the mapping $R \mapsto glb(R)$ is an abstraction operation summarizing a region into a ternary vector. Conversely, any ternary vector $\mathbf{b} \in \mathbb{S}^l$ naturally corresponds to a region $set(\mathbf{b}) \subseteq \mathbb{B}^l$ given as $set(\mathbf{b}) =_{df} \{x \mid \mathbf{b} \sqsubseteq x\} \subseteq \mathbb{B}^l$. The mappings (glb, set) form what is known as an *abstraction-refinement* or *projection-embedding* pair (see, e.g., [11]), satisfying $R \subseteq set(glb(R))$ and $\mathbf{b} = glb(set(\mathbf{b}))$. Note that $R \subseteq set(glb(R))$ is an in-equation is because we lose information, in general, when we turn a region into a ternary vector and then try to reconstruct the region from its abstraction.[6] Only for cubical regions (those that can be coded as min-terms) can we break up the logical constraint contained in them in terms of a constraint on individual signal values expressible by ternary vectors. More specifically, we have the following lemma:

[6]Because of the different cardinalities of $2^{\mathbb{B}^l}$ (set of regions in $\mathbb{B}^l$) and $\mathbb{S}^l$ there is no way in which we could use ternary vectors to represent arbitrary binary regions in a one-to-one fashion.

Lemma 2 *Let $R \subseteq \mathbb{B}^l$ be a non-empty region and $\mathbf{r} = glb(R) \in \mathbb{S}^l$ its ternary abstraction. Then, the following are equivalent*:

- *R is cubical*
- *$R = set(\mathbf{r})$*
- *For all $x \in \mathbb{B}^l$, $x \in R$ iff $\mathbf{r} \sqsubseteq x$.*

The equivalence $x \in R$ iff $\mathbf{r} \sqsubseteq x$ means that the ternary vector $\mathbf{r} = glb(R)$ contains the same information as the original cubical region R, except that the set-theoretic condition $x \in R$ has been replaced by an algebraic condition $\mathbf{r} \sqsubseteq x$. It is this reduction of a logical problem of exponential nature into an algebraic one of linear nature that is the key selling point of ternary simulation.

Let us now look more closely at the algebraic properties of the domain $\mathbb{TS} = (\mathbb{B} \times \mathbb{R}^+)_\perp$ of timed ternary values. It is equipped with a natural partial ordering $\sqsubseteq$ such that $\perp \sqsubseteq (\alpha, \tau)$ and $(\alpha_1, \tau_1) \sqsubseteq (\alpha_2, \tau_2)$ iff $\alpha_1 = \alpha_2$ and $\tau_2 \leq \tau_1$. Intuitively, $\mathbf{x} \sqsubseteq \mathbf{y}$ means the value $\mathbf{y}$ is more defined and earlier than $\mathbf{x}$. The totally undefined value $\perp \in \mathbb{TS}$ may be identified with the pair $(\perp, \infty)$, thinking of $\mathbb{TS}$ as the coalesced[7] product $\mathbb{S} \otimes \mathbb{R}^+_\infty$ of domains, where $\mathbb{R}^+_\infty = \mathbb{R}^+ \cup \{\infty\}$.

Let us call a subset $\mathbf{X} \subseteq \mathbb{TS}$ *consistent* if there exists $\mathbf{z} \in \mathbb{TS}$ such that $\forall \mathbf{x} \in \mathbf{X}$, $\mathbf{x} \sqsubseteq \mathbf{z}$. Concretely, $\mathbf{X}$ is consistent iff it does not contain opposite Boolean values, i.e., $\forall (\alpha_1, \tau_1), (\alpha_2, \tau_2) \in \mathbf{X}$ we have $\alpha_1 = \alpha_2$. In particular, every chain $\mathbf{x}_1 \sqsubseteq \mathbf{x}_2 \sqsubseteq \cdots$ is consistent. The domain $(\mathbb{TS}, \sqsubseteq)$ is *consistent complete*, i.e., every consistent subset $\mathbf{X} \subseteq \mathbb{TS}$ has a least upper bound $\bigvee \mathbf{X}$. For consistent $\mathbf{X} = \{(\alpha, \tau_1), (\alpha, \tau_2), \ldots, (\alpha, \tau_l)\}$ we find $\bigvee \mathbf{X} = (\alpha, min\{\tau_i \mid i \leq l\})$.

In the binary case we write $\mathbf{x} \sqcup \mathbf{y}$ instead of $\bigvee\{\mathbf{x}, \mathbf{y}\}$. The ordering $\sqsubseteq$ is lifted to vectors $\mathbb{TS}^l$ in the component-wise fashion, so that $(\mathbb{TS}^l, \sqsubseteq)$ is a consistent complete partial ordering (see [11]).

On $\mathbb{TS}$ we define two time-related operations. The first is *time projection*, translating $\mathbf{x} \in \mathbb{TS}$ and $E \in \mathbb{R}^+_\infty$ to the ternary vector $\mathbf{x}|_E \in \mathbb{S}$ putting $(\alpha, \tau)|_E = \alpha$ if $\tau \leq E$ and $\mathbf{x}|_E = \perp$ in all other cases. The time projection $\mathbf{x}|_E$ represents the value $\mathbf{x}$ "by time" E, i.e., interpreting $\mathbf{x}$ as undefined if it does not stabilize before or at E time. The special case $\mathbf{x}|_\infty$ simply removes the time parameter and thus corresponds to the stationary abstraction. Time projection can also be applied to *timed ternary vectors* $\mathbf{x} \in \mathbb{TS}^l$, so that $\langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l \rangle|_E = \langle \mathbf{x}_1|_E, \mathbf{x}_2|_E, \ldots, \mathbf{x}_l|_E \rangle$. Obviously, time projection is monotonic, i.e., if $\mathbf{x} \sqsubseteq \mathbf{y}$ in $\mathbb{TS}^l$ then $\mathbf{x}|_E \sqsubseteq \mathbf{y}|_E$ in $\mathbb{S}^l$. Another useful operation is the *time delay*, taking $\mathbf{x} \in \mathbb{TS}$ and $E \in \mathbb{R}^+$ to the delayed value $\mathbf{x} + E \in \mathbb{TS}$ defined as $(\alpha, \tau) + E = (\alpha, \tau + E)$ and $\perp + E = \perp$. This lifts to timed ternary vectors $\mathbf{x} \in \mathbb{TS}^l$ and vectors of delays $\mathbf{E} \in (\mathbb{R}^+)^l$ in the obvious way, i.e., $\mathbf{x} + \mathbf{E}$ adds the delay E_i to component $\mathbf{x}_i$. Time delay, too, is monotonic.

The timed ternary domain $\mathbb{TS}^l$ is a refinement of the ternary domain $\mathbb{S}^l$. Time projection $\mathbf{x}|_\infty$ provides a way of abstracting away the extra temporal information. It is also possible to go the other way, embedding $\mathbb{S}^l$ into $\mathbb{TS}^l$. Every ternary vector $\mathbf{x} \in \mathbb{S}^l$ can be turned naturally into a timed ternary vector $val(\mathbf{x}) \in \mathbb{TS}^l$ so that $val_i(\mathbf{x}) = (\alpha, 0)$ if $\mathbf{x}_i = \alpha \in \mathbb{B}$ and $val_i(\mathbf{x}) = \perp$ if $\mathbf{x}_i = \perp$. Every monotonic ternary function $\mathbf{F} : \mathbb{S}^{m+n} \to \mathbb{S}^m$ can be extended to a monotonic function $let(\mathbf{F}) : \mathbb{TS}^{m+n} \to \mathbb{TS}^n$, the *timed ternary extension*, in a natural way, putting $(i \leq n)$

$$let_i(\mathbf{F})(\mathbf{x}) =_{df} \big(\mathbf{F}_i(\mathbf{x}|_\infty), min\big\{E \mid \mathbf{F}_i(\mathbf{x}|_E) \neq \perp\big\}\big). \tag{5}$$

[7]The term "coalesced" means that all elements $(x, y) \in \mathbb{S} \otimes \mathbb{R}^+_\infty$ in which $x = \perp$ or $y = \infty$ are identified.

Intuitively, $let_i(\mathbf{F})(\mathbf{x})$ returns in its value component the stationary value of $\mathbf{F}_i$ under $\mathbf{x}$, and in the time component the earliest stabilization time for this value, viz. the minimal E such that $\mathbf{F}_i(\mathbf{x}|_E)$ is different from $\bot$. This minimum is well-defined since the set of ternary inputs scanned, $\{\mathbf{x}|_E \mid E \in \mathbb{R}^+\} \subseteq \mathbb{S}^{m+n}$, is necessarily finite and since $\mathbf{x}|_\tau$ as a function of τ can change value only a finite number of times. Note that since $min\,\emptyset = \infty$ by mathematical convention, we obtain $let_i(\mathbf{F})(\mathbf{x}) = (\bot, \infty) = \bot$ in the special case where $\mathbf{F}_i(\mathbf{x}|_\infty) = \bot$. When $let_i(\mathbf{F})(\mathbf{x}) = (\alpha, \tau)$ for $\alpha \in \mathbb{B}$ then $\mathbf{F}_i(\mathbf{x}|_\infty) = \alpha$ and $\tau = min\{E \mid \mathbf{F}_i(\mathbf{x}|_E) = \alpha\}$.

Example 9 Take the Boolean function $S : \mathbb{B}^3 \to \mathbb{B}$ specified $S\langle x, s_1, s_2\rangle = \overline{x \cdot s_1} + s_2$. Its ternary extension $\mathbf{S} : \mathbb{S}^3 \to \mathbb{S}$ is $\mathbf{S}(\mathbf{z}) = glb(S(set(\mathbf{z})))$. Let us study $\mathbf{S}$ under three ternary input stimuli $\mathbf{z}_1 = 1\bot\bot$, $\mathbf{z}_2 = \bot 01$ and $\mathbf{z}_3 = 110$. The first of these represents the region $set(\mathbf{z}_1) = set(1\bot\bot) = \{100, 101, 110, 111\}$. Evaluating the function S for each element separately yields $S(set(\mathbf{z}_1)) = \{\overline{1\cdot 0} + 0, \overline{1\cdot 0} + 1, \overline{1\cdot 1} + 0, \overline{1\cdot 1} + 1\} = \{1, 0\}$. For $\mathbf{z}_2 = \bot 01$ we get $S(set(\mathbf{z}_2)) = S(\{001, 101\}) = \{\overline{0\cdot 0} + 1, \overline{1\cdot 0} + 1\} = \{1\}$ and for $\mathbf{z}_3 = 110$, $S(set(\mathbf{z}_3)) = S(\{110\}) = \{\overline{1\cdot 1} + 0\} = \{0\}$. In the ternary extension $\mathbf{S}$ these sets are approximated using glb, i.e., $\mathbf{S}(\mathbf{z}_1) = glb(S(set(\mathbf{z}_1))) = glb(\{1, 0\}) = \bot$, $\mathbf{S}(\mathbf{z}_2) = glb(\{1\}) = 1$ and $\mathbf{S}(\mathbf{z}_3) = glb(\{0\}) = 0$. Observe that the expression $\overline{x \cdot s_1} + s_2$ yields value 1 as soon as either $s_1 = 0$ or $s_2 = 1$, i.e., in $\mathbf{z}_2 = \bot 01$ both components 0 and 1 are "controlling" values. Hence, already the approximations $\bot 0\bot$, $\bot\bot 1 \sqsubseteq \mathbf{z}_2$ force the output $\mathbf{S}(\bot 0\bot) = \mathbf{S}(\bot\bot 1) = 1$. This is not the case of the totally undefined $\bot\bot\bot \sqsubseteq \mathbf{z}_2$ since $\mathbf{S}(\bot\bot\bot) \sqsubseteq \mathbf{S}(\mathbf{z}_1) = \bot$ by monotonicity of $\mathbf{S}$. For input $\mathbf{z}_3 = 110$ all three values are needed to drive the output high, so $\mathbf{S}(\mathbf{z}) = \bot$ for all $\mathbf{z} \sqsubseteq 110$ with $\mathbf{z} \neq 110$.

The ternary vectors $\mathbf{z}_1$, $\mathbf{z}_2$, $\mathbf{z}_3$ can be extended to timed ternary vectors by adding a delay value to every component that is different from $\bot$. For instance, $\mathbf{x}_1 = (1, D_1)\bot\bot$, $\mathbf{x}_2 = \bot(0, D_1)(1, D_2)$ and $\mathbf{x}_3 = (1, 0)(1, D_1)(0, D_2)$ are such extensions. They encode timed stabilization functions in the sense that they determine, for every time τ, a ternary vector $\mathbf{x}_i|_\tau \in \mathbb{S}$. Specifically, $\mathbf{x}_1|_\tau = ((1, D_1)\bot\bot)|_\tau = (1, D_1)|_\tau \bot|_\tau \bot|_\tau = \bot\bot\bot$ for all $\tau < D_1$ and $\mathbf{x}_1|_\tau = 1\bot\bot$ if $\tau \geq D_1$. In other words, $\mathbf{x}_1$ before time D_1 looks just like the fully undefined vector $\bot\bot\bot$, but from time D_1 onwards its first component is stable at 1. For $\mathbf{x}_2$ (analogously for $\mathbf{x}_3$) we have

$$\mathbf{x}_2|_\tau = \begin{cases} \bot\bot\bot & \text{if } \tau < min(D_1, D_2) \\ \bot 0\bot & \text{if } D_1 \leq \tau < D_2 \\ \bot\bot 1 & \text{if } D_2 \leq \tau < D_1 \\ \bot 0 1 & \text{if } \tau \geq max(D_1, D_2) \end{cases}$$

which codes a somewhat richer switching: Before time $min(D_1, D_2)$, $\mathbf{x}_2$ behaves like $\bot\bot\bot$. Then, depending on whether (i) D_1 or (ii) D_2 is smaller, in the interval between these two bounds either (i) the second component becomes 0 or (ii) the third component stabilizes to 1. Only from $max(D_1, D_2)$ onwards are we guaranteed the stationary value $\bot 0 1$. Note that the steady state abstractions $\mathbf{x}_i|_\infty = \mathbf{z}_i$ in all cases are the original ternary vectors.

Finally we consider the timed extension $let(\mathbf{S}) : \mathbb{TS}^3 \to \mathbb{TS}$ of $\mathbf{S}$. By Definition 5, for each input $\mathbf{x}$, $let(\mathbf{S})(\mathbf{x})$ produces the stationary output value $\mathbf{S}(\mathbf{x}|_\infty)$ together with its earliest stabilization time $min\{E \mid \mathbf{S}(\mathbf{x}|_E) \neq \bot\}$. For instance, we have $\mathbf{S}(\mathbf{x}_1|_\infty) = \mathbf{S}(\mathbf{z}_1) = \bot$ and hence $min\{E \mid \mathbf{S}(\mathbf{x}_1|_E) \neq \bot\} = \infty$, whence $let(\mathbf{S})(\mathbf{x}_1) = (\bot, \infty) = \bot$. For $\mathbf{x}_2$, in contrast, $\mathbf{S}(\mathbf{x}_2|_\infty) = \mathbf{S}(\mathbf{z}_2) = 1$ and $min\{E \mid \mathbf{S}(\mathbf{x}_2|_E) \neq \bot\} = \min(D_1, D_2)$ since *either* $\bot 0\bot$ (which has delay D_1) or $\bot\bot 1$ (with delay D_2) drives the output of S high and thus $\mathbf{S}(\mathbf{x}_2|_{min(D_1,D_2)}) = 1 \neq \bot$. Finally, consider $\mathbf{x}_3$ for which $\mathbf{S}(\mathbf{x}_3|_\infty) = \mathbf{S}(\mathbf{z}_3) = 0$ but now $min\{E \mid \mathbf{S}(\mathbf{x}_3|_E) = 0 \neq \bot\} = max(D_1, D_2)$ because we must wait for all components of $\mathbf{x}_3$ to stabilize in order to force S to show output 1.

3.3.2 The simulation algorithm

We finally put timed ternary vectors to work. Let the specification of a (well-formed) network N be given by a rsnf theory $\Phi_N = \{s_i :=_{D_i} S_i \mid i \leq m\}$ in state vertices s_i and associated excitation functions S_i, $i \leq m$. Let the input be $\mathcal{X} \equiv a$ and the network specification $\Phi_{N,a} = \Phi_N, \Diamond_0(\mathcal{X} \equiv a)$. The simulation algorithm builds a sequence $\mathbf{s}^0, \mathbf{s}^1, \ldots, \mathbf{s}^h \in \mathbb{TS}^m$ of timed ternary vectors to approximate the network's provable stabilization behavior in the sense that for all $\tau \geq 0$ and $R \subseteq \mathbb{B}^m$ we have $\Phi_{N,a} \vdash \Diamond_\tau R$ iff there exists h such that $set(\mathbf{s}^h|_\tau) \subseteq R$. To this end let $\mathbf{S} : \mathbb{S}^{m+n} \to \mathbb{S}^m$ be the ternary extension of the total system excitation $S : \mathbb{B}^{m+n} \to \mathbb{B}^m$. The *timed ternary simulation* of N under $\mathcal{X} \equiv a$ proceeds as follows:

$$\mathbf{s}^0 =_{df} \bot^m \tag{6}$$

$$\mathbf{s}^{h+1} =_{df} \mathbf{N}\big(val(a).\mathbf{s}^h\big) \quad \text{where } \mathbf{N}(\mathbf{z}) =_{df} let(\mathbf{S})(\mathbf{z}) + \mathbf{D}. \tag{7}$$

In this expression, $val(a).\mathbf{s}^h$ is the concatenation of the input vector $val(a) \in \mathbb{TS}^n$ with state vector $\mathbf{s}^h \in \mathbb{TS}^m$ to give a total state in $\mathbb{TS}^{m+n}$. The function $\mathbf{N} : \mathbb{TS}^{m+n} \to \mathbb{TS}^m$ defined in (7) is the *timed ternary switching function* of N. The term $\mathbf{N}(val(a).\mathbf{s}^h)$ computes an updated stabilization time and stabilization value for every state signal s_i from the input $val(a)$ and the previous approximation $\mathbf{s}^h$ of the state vector. This switching function is obtained by taking the *timed ternary excitation* $let(\mathbf{S})(\mathbf{z})$ and adding to it the system delay $\mathbf{D}$. This corresponds to the switching of all the delay elements and updating of all state vertices with their new stabilization values. Since the domain $\mathbb{TS}^m$ is a consistent complete partial ordering and $\mathbf{N}$ monotonic on $\mathbb{TS}^m$, the iteration (6) and (7) becomes stationary and constructs the *timed ternary response function* $[\![N]\!] : \mathbb{B}^n \to \mathbb{TS}^m$ as the least fixed point $[\![N]\!](a) =_{\mathrm{df}} \mu\mathbf{y}.\mathbf{N}(val(a).\mathbf{y})$.

Example 10 Consider network N_2 from Fig. 10 with $\mathcal{X} = \{x\}$, $\mathcal{S} = \{s_1, s_2\}$ and rsnf specification $\Phi_{N_2} = \{s_1 :=_{D_1} \overline{x}, s_2 :=_{D_2} \overline{x \cdot s_1} + s_2\}$ whose symbolic simulation in UN-calculus we have discussed in Sect. 3.2. Now we repeat this simulation using timed ternary vectors. The total system excitation function is $S : \mathbb{B}^3 \to \mathbb{B}^2$ such that $S\langle x, s_1, s_2\rangle = \langle S_1, S_2\rangle = \langle \overline{x}, \overline{x \cdot s_1} + s_2\rangle$. The ternary abstraction $\mathbf{S} : \mathbb{S}^3 \to \mathbb{S}^2$ of S is given as

$$\begin{aligned}\mathbf{S}\langle \mathbf{x}, \mathbf{s}_1, \mathbf{s}_2\rangle &= \langle \mathbf{S}_1, \mathbf{S}_2\rangle \\ &= \big\langle glb\big(S_1\big(set\langle \mathbf{x}, \mathbf{s}_1, \mathbf{s}_2\rangle\big)\big), glb\big(S_2\big(set\langle \mathbf{x}, \mathbf{s}_1, \mathbf{s}_2\rangle\big)\big)\big\rangle\end{aligned}$$

and the timed ternary excitation $let(\mathbf{S}) : \mathbb{TS}^3 \to \mathbb{TS}^2$, which enriches $\mathbf{S}$ by quantitative delay information, is $let(\mathbf{S})(\mathbf{z}) = \langle let_1(\mathbf{S})(\mathbf{z}), let_2(\mathbf{S})(\mathbf{z})\rangle$ where

$$\begin{aligned}let_1(\mathbf{S})(\mathbf{z}) &= \big(\mathbf{S}_1(\mathbf{z}|_\infty), min\big\{E \mid \mathbf{S}_1(\mathbf{z}|_E) \neq \bot\big\}\big) \\ let_2(\mathbf{S})(\mathbf{z}) &= \big(\mathbf{S}_2(\mathbf{z}|_\infty), min\big\{E \mid \mathbf{S}_2(\mathbf{z}|_E) \neq \bot\big\}\big).\end{aligned}$$

The system's switching function $\mathbf{N}(\mathbf{z}) =_{df} \langle let_1(\mathbf{S})(\mathbf{z}) + D_1, let_2(\mathbf{S})(\mathbf{z}) + D_2\rangle$ adds to this the system delays $\mathbf{D} = \langle D_1, D_2\rangle$ at each iteration step (7).

Ternary simulation of N_2 for input $x \equiv a$ starts with (6) in the timed ternary state $\mathbf{s}^0 = \bot\bot \in \mathbb{TS}^2$, in which no assumptions regarding value or timing are made. The state information $\mathbf{s}^0 = \bot\bot$ is completed by the input a lifted as a timed ternary value $val(a) = (a, 0)$, giving $\mathbf{z}^0 = val(a).\mathbf{s}^0 = (a, 0)\bot\bot$. This is the initial approximation of the *total* state of N_2.

It represents a functional stabilization bound in the sense that it determines, for every time τ, a ternary vector $\mathbf{z}^0|_\tau$ such that N_2 is guaranteed to be in state region $set(\mathbf{z}^0|_\tau) \subseteq \mathbb{B}^S$ by time τ. In this case, $\mathbf{z}^0|_\tau = ((a,0) \perp \perp)|_\tau = (a,0)|_\tau \perp|_\tau \perp|_\tau = a \perp \perp$ for all $\tau \geq 0$. Iterating (7) we send $\mathbf{z}^0$ through the system switching function to generate the fixed point approximation sequence $\mathbf{s}^{h+1} = \mathbf{N}(\mathbf{z}^h)$, $\mathbf{z}^h = val(a).\mathbf{s}^h$.

For input $a = 0$ we find that $\mathbf{s}^{h+1}$ is constant $\mathbf{s}^{h+1} = (1, D_1)\,(1, D_2)$ for all $h \geq 0$. Thus the fixed point is reached after the first iteration step already. This means that state signals s_1, s_2 will stabilize at 1 by time D_1 and D_2, respectively. To show that $\mathbf{s}^{h+1} = (1, D_1)\,(1, D_2)$ we first determine the two state components $let_1(\mathbf{S})(\mathbf{z}^h)$ and $let_2(\mathbf{S})(\mathbf{z}^h)$ separately:

- $let_1(\mathbf{S})(\mathbf{z}^h)$ is a timed ternary vector with value $\mathbf{S}_1(\mathbf{z}^h|_\infty)$ and delay bound $min\{E \mid \mathbf{S}_1(\mathbf{z}^h|_E) \neq \perp\}$. Since $\mathbf{z}^h|_\tau = (val(a).\mathbf{s}^h)|_\tau = a.\mathbf{s}^h|_\tau = 0\,\mathbf{s}_1^h|_\tau\,\mathbf{s}_2^h|_\tau$ we get $\mathbf{S}_1(\mathbf{z}^h|_\tau) = \mathbf{S}_1(0\,\mathbf{s}_1^h|_\tau\,\mathbf{s}_2^h|_\tau) = glb(S_1(set(0\,\mathbf{s}_1^h|_\tau\,\mathbf{s}_2^h|_\tau))) = glb\,\{\overline{x} \mid \langle x, s_1, s_2\rangle \in set(0\,\mathbf{s}_1^h|_\tau\,\mathbf{s}_2^h|_\tau)\} = glb\,\{\overline{0}\} = 1$ for all $\tau \geq 0$. Hence $\mathbf{S}_1(\mathbf{z}^h|_\infty) = 1$ and $min\{E \mid \mathbf{S}_1(\mathbf{z}^h|_E) \neq \perp\} = 0$ which means $let_1(\mathbf{S})(\mathbf{z}^h) = (1, 0)$.
- To obtain the second component $let_2(\mathbf{S})(\mathbf{z}^h)$ we look at the evaluation of $\mathbf{S}_2(\mathbf{z}^h|_\tau)$ as a function of τ. Here we obtain $\mathbf{S}_2(\mathbf{z}^h|_\tau) = \mathbf{S}_2(0\,\mathbf{s}_1|_\tau\,\mathbf{s}_2|_\tau) = glb(S_2(set(0\,\mathbf{s}_1|_\tau\,\mathbf{s}_2|_\tau))) = glb\,\{\overline{x \cdot s_1} + s_2 \mid \langle x, s_1, s_2\rangle \in set(0\,\mathbf{s}_1|_\tau\,\mathbf{s}_2|_\tau)\} = glb\,\{1\} = 1$. From this it follows that $\mathbf{S}_2(\mathbf{z}^h|_\infty) = 1$ and $min\{E \mid \mathbf{S}_1(\mathbf{z}^h|_E) \neq \perp\} = 0$ so that $let_2(\mathbf{S})(\mathbf{z}^h)) = (1, 0)$, overall.

Summing up, we have found $\langle let_1(\mathbf{S})(\mathbf{z}^h), let_2(\mathbf{S})(\mathbf{z}^h)\rangle = (1,0)(1,0)$, which tells us that at time 0 both excitation functions S_1 and S_2 evaluate to stable value 1. In the symbolic simulation of N_2 in Sect. 3.2 this corresponds to the derivation of $\Diamond_0(S_1 \equiv 1)$ and $\Diamond_0(S_2 \equiv 1)$ from the hypothesis $\Diamond_0(x \equiv 0)$ using rule $\Diamond id$.

Next, we perform the switching of state nodes based on this excitation information. This is done by computing $\mathbf{s}^{h+1} = \mathbf{N}(\mathbf{z}^h) = \langle let_1(\mathbf{S})(\mathbf{z}^h) + D_1, let_2(\mathbf{S})(\mathbf{z}^h)) + D_2\rangle = \langle (1,0) + D_1, (1,0) + D_2\rangle = (1, D_1)(1, D_2)$ as claimed above. In the symbolic simulation this corresponds to deriving $\Diamond_{D_1}(s_1 \equiv 1)$ and $\Diamond_{D_2}(s_2 \equiv 1)$ by way of rule $\supset\Diamond L$ from $\Diamond_0(S_1 \equiv 1)$ and $\Diamond_0(S_2 \equiv 1)$, respectively, together with the network specifications $s_1 :=_{D_1} S_1$ and $s_2 :=_{D_2} S_2$.

Now let us simulate N_2 with input $a = 1$, initializing with $\mathbf{z}^0 = val(a).\mathbf{s}^0 = (1,0) \perp \perp$. Here we find $\mathbf{z}^0|_\tau = 1 \perp \perp$ and $\mathbf{S}_1(\mathbf{z}^0|_\tau) = \mathbf{S}_1(1 \perp \perp) = glb(S_1(set(1 \perp \perp))) = glb\,\{\overline{x} \mid \langle x, s_1, s_2\rangle \in set(1 \perp \perp)\} = glb\,\{0\} = 0$ for all $\tau \geq 0$. From this we get $let_1(\mathbf{S})(\mathbf{z}^0) = (0,0)$ for state s_1. For the second state vertex, $\mathbf{S}_2(\mathbf{z}^0|_\tau) = \mathbf{S}_2(1 \perp \perp) = glb(S_2(set(1 \perp \perp))) = glb\,\{\overline{x \cdot s_1} + s_2 \mid \langle x, s_1, s_2\rangle \in set(1 \perp \perp)\} = glb\,\{0, 1\} = \perp$, whence $let_2(\mathbf{S})(\mathbf{z}^0) = \perp$. Then, the state update yields $\mathbf{s}^1 = \mathbf{N}(\mathbf{z}^0) = \langle let_1(\mathbf{S})(\mathbf{z}^0) + D_1, let_2(\mathbf{S})(\mathbf{z}^0) + D_2\rangle = \langle (0,0) + D_1, \perp + D_2\rangle = (0, D_1) \perp$. This agrees with the symbolic simulation where we derived $\Diamond_{D_1}(s_1 \equiv 0)$.

The fact that we also have $\Diamond_{D_1+D_2}(s_2 \equiv 1)$ is not visible yet. For state vertex s_2 we still have no information in $\mathbf{s}^1 = \langle \mathbf{s}_1^1, \mathbf{s}_2^1\rangle = (0, D_1) \perp$. We must go through another approximation round: The first component of the excitation remains the same $let_1(\mathbf{S})(\mathbf{z}^1) = let_1(\mathbf{S})(val(1).\mathbf{s}^1) = (0,0)$ but the second $let_2(\mathbf{S})(\mathbf{z}^1)$ tightens up: $\mathbf{z}^1|_\tau = (val(1).\mathbf{s}^1)|_\tau = ((1,0)\,(0, D_1) \perp)|_\tau = 1 \perp \perp$ for $0 \leq \tau < D_1$ and $\mathbf{z}^1|_\tau = 1\,0 \perp$ for $\tau \geq D_1$. In the first case, $0 \leq \tau < D_1$, we have $\mathbf{S}_2(\mathbf{z}^1|_\tau) = \mathbf{S}_2(1 \perp \perp) = \perp$ as before, while for $\tau \geq D_1$ we get $\mathbf{S}_2(\mathbf{z}^1|_\tau) = \mathbf{S}_2(1\,0 \perp) = glb\,\{\overline{x \cdot s_1} + s_2 \mid \langle x, s_1, s_2\rangle \in set(1\,0 \perp)\} = glb\,\{1\} = 1$. Therefore, $let_2(\mathbf{S})(\mathbf{z}^1) = (1, D_1)$ which shows that signal s_2 is bound to stabilize at 1 by time D_1. This aligns with the symbolic simulation which obtains $\Diamond_{D_1}(\overline{x \cdot s_1} + s_2 \equiv 1)$, or equivalently $\Diamond_{D_1}(S_2 \equiv 1)$, from $\Diamond_{D_1}(s_1 \equiv 0)$ using rule $\Diamond id$. Updating the state vector $\mathbf{s}^2 = \mathbf{N}(\mathbf{z}^1) = \langle let_1(\mathbf{S})(\mathbf{z}^1) + D_1, let_2(\mathbf{S})(\mathbf{z}^1) + D_2\rangle = \langle (0,0) + D_1, (1, D_1) + D_2\rangle = (0, D_1)(1, D_1 + D_2)$ corresponds to deriving $\Diamond_{D_1+D_2}(s_2 \equiv 1)$ from $\Diamond_{D_1}(S_2 \equiv 1)$ invoking rule $\supset\Diamond L$ and $s_2 :=_{D_1} S_2$. We now reached the fixed point for we find $\mathbf{s}^3 = let(\mathbf{S})(\mathbf{z}^2) + \mathbf{D} = (0, D_1)\,(1, D_1 + D_2) = \mathbf{s}^2$.

In sum, we have determined the timed ternary response $[\![N_2]\!] : \mathbb{B} \to \mathbb{TS}^2$, viz. $[\![N_2]\!](0) = (1, D_1)(1, D_2)$ and $[\![N_2]\!](1) = (1, D_1)(1, D_1 + D_2)$. The fact that $[\![N_2]\!](a) \neq \bot$ for all a means that N_2 is ternary combinational. The steady state $[\![N_2]\!](a)|_\infty$ is identical $(1, 1)$ for all inputs $a \in \mathbb{B}$ but the stabilization time varies: State vertex s_1 has bound D_1 for all inputs, while s_2 is known to stabilize within D_1 only for input $a = 0$; For $a = 1$ the vertex s_2 may take up to $D_1 + D_2$ time units.

We are now ready to show that ternary simulation is a sound and complete decision procedure for the derivability of timed regions $\Diamond_D R$ from network theories $\Phi_{N,a}$. It gives a precise logical interpretation of the fixed point response $[\![N]\!](a) \in \mathbb{TS}^m$: If $[\![N]\!]_i(a) = (\alpha, \tau)$, then α is the unique value and τ the minimal time such that $\Diamond_\tau \langle s_i \equiv \alpha \rangle$ can be derived from $\Phi_{N,a}$. If $[\![N]\!]_i(a) = \bot$ then there is no delay E and value α such that $\Diamond_E \langle s_i \equiv \alpha \rangle$ is derivable from $\Phi_{N,a}$.

Theorem 5 (Computational adequacy of timed ternary simulation) *Let* $[\![N]\!](a) =_{df} \mu \mathbf{y}.\mathbf{N}(val(a).\mathbf{y})$ *be the response function of network* N *computed by* (6) *and* (7). *Then, for every region* $R \subseteq \mathbb{B}^m$, *input* $a \in \mathbb{B}^n$ *and* $\tau \geq 0$ *we have*

$$set\big([\![N]\!](a)|_\tau\big) \subseteq R \quad \textit{iff } \Phi_{N,a} \vdash \Diamond_\tau R.$$

Proof Throughout the proof we will keep the input $a \in \mathbb{B}^n$ fixed. We actually prove the following somewhat more general statement

$$set\big(a.[\![N]\!](a)|_\tau\big) \subseteq R \quad \text{iff } \Phi_{N,a} \vdash \Diamond_\tau R, \tag{8}$$

in which $R \subseteq \mathbb{B}^{m+n}$ may be an *arbitrary* region of the total state space rather than just a region in the state space $\mathbb{B}^m$. The statement of Theorem 5 can be obtained as a special case observing that $set(a.[\![N]\!](a)|_\tau) = (\mathcal{X} \equiv a) \wedge set([\![N]\!](a)|_\tau)$ and $((\mathcal{X} \equiv a) \wedge S) \subseteq R$ iff $S \subseteq R$ whenever $R \subseteq \mathbb{B}^m$ is a region constraining only state vertices.

Moreover, it will be convenient to extend the special equivalences $\phi \equiv 0$, $\phi \equiv 1$ where 0, 1 are identified with *false* and *true*, respectively, to permit the ternary value $\bot$ as well. Specifically, let us consider $\phi \equiv \bot$ as an abbreviation[8] for $(\mathit{false} \supset \phi) \wedge (\phi \supset \mathit{true})$, or, equivalently as an abbreviation for 1. With this convention one can show that for every $\mathbf{s} = \langle s_1, s_2, \ldots, s_m \rangle \in \mathbb{S}^m$ and $\tau \geq 0$

$$\Phi_{N,a} \vdash \Diamond_\tau set(a.\mathbf{s}) \iff \forall i \leq m.\ \Phi_{N,a} \vdash \Diamond_\tau (s_i \equiv \mathbf{s}_i). \tag{9}$$

This follows from $\Phi_{N,a} \models \Diamond_0(\mathcal{X} \equiv a)$ by way of rules $\Diamond \wedge R$, $\Diamond id$ and the fact that $set(a.\mathbf{s})$ is cubical, specifically that $set(a.\mathbf{s}) = (\mathcal{X} \equiv a) \wedge \bigwedge_{i \leq m} (s_i \equiv \mathbf{s}_i)$.

For direction $(\Rightarrow)$ of (8) it suffices to show that $\Phi_{N,a} \vdash \Diamond_\tau set(a.[\![N]\!](a)|_\tau)$ because of rule $\Diamond id$. Now recall that $[\![N]\!](a)$ is the fixed point of a finite approximation sequence $\mathbf{s}^0, \mathbf{s}^1, \ldots, \mathbf{s}^h \in \mathbb{TS}^m$ defined by (6) and (7) with limit $\mathbf{s}^\infty = [\![N]\!](a)$. We show by induction on h that $\Phi_{N,a} \vdash \Diamond_\tau set(a.\mathbf{s}^h|_\tau)$. Since $set(a.\mathbf{s}^\infty|_\tau) = set(a.[\![N]\!](a)|_\tau)$ we are done.

For the base case $h = 0$ we have $\mathbf{s}^0 = \bot^m$ and $set(a.\mathbf{s}^0|_\tau) = (\mathcal{X} \equiv a) \wedge \bigwedge_{i \leq m} (s_i \equiv \bot) = \mathcal{X} \equiv a$, whence trivially $\Phi_{N,a} \vdash \Diamond_\tau set(a.\mathbf{s}^0|_\tau)$ by rule $\Diamond id$ and since $\Diamond_0(\mathcal{X} \equiv a)$ is contained in $\Phi_{N,a}$.

[8] In other words, $\bot$ is treated as the completely "uninformative" truth value which is both 0 and 1 at the same time. Interestingly, the missing fourth value $\top$ for "inconsistency" arises in a dual fashion by reading $\phi \equiv \top$ as an abbreviation for $(\mathit{true} \supset \phi) \wedge (\phi \supset \mathit{false})$. Obviously, this is a contradiction and equivalent to *false*.

For the induction step we claim that $\Phi_{N,a} \vdash \Diamond_\tau set(a.\mathbf{s}^{h+1}|_\tau)$. Now we exploit (9) which reduces the problem to showing

$$\forall i \leq m.\ \Phi_{N,a} \vdash \Diamond_\tau \big(s_i \equiv \mathbf{s}_i^{h+1}|_\tau\big).$$

Fix any such state index $i \leq m$. Recall that the iteration value is computed as $\mathbf{s}_i^{h+1} = let_i(\mathbf{S})(\mathbf{x}^h) + D_i$ where $\mathbf{x}^h = val(a).\mathbf{s}^h$ and the timed ternary lifting is given in (5)

$$let_i(\mathbf{S})\big(\mathbf{x}^h\big) = \big(\mathbf{S}_i\big(\mathbf{x}^h|_\infty\big), min\big\{E \mid \mathbf{S}_i\big(\mathbf{x}^h|_E\big) \neq \bot\big\}\big). \tag{10}$$

If $\mathbf{S}_i(\mathbf{x}^h|_\infty) = \bot$ then also $let_i(\mathbf{S})(\mathbf{x}^h) = (\bot, \infty) = \bot$ and thus $\mathbf{s}_i^{h+1}|_\tau = (let_i(\mathbf{S})(\mathbf{x}^h) + D_i)|_\tau = (\bot + D_i)|_\tau = \bot$, so that $(s_i \equiv \mathbf{s}_i^{h+1}|_\tau) = (s_i \equiv \bot) = 1$. Hence, we are trivially done because $\Phi_{N,a} \vdash \Diamond_\tau(s_i \equiv \bot)$ is the same as $\Phi_{N,a} \vdash \Diamond_\tau 1$ which is derivable by rule $\Diamond true$. So, let us assume henceforth that $\mathbf{S}_i(\mathbf{x}^h|_\infty) = \alpha \neq \bot$ and $D = min\{E \mid \mathbf{S}_i(\mathbf{x}^h|_E) = \alpha\}$, i.e., $\mathbf{s}_i^{h+1} = let_i(\mathbf{S})(\mathbf{x}^h) + D_i = (\alpha, D + D_i)$. Again, matters are trivial if $\tau < D + D_i$ because then $\mathbf{s}_i^{h+1}|_\tau = (\alpha, D + D_i)|_\tau = \bot$, so that $\Phi_{N,a} \vdash \Diamond_\tau(s_i \equiv \mathbf{s}_i^{h+1}|_\tau)$ is trivially derivable like before. The interesting case is where $\tau \geq D + D_i$ implying that $\mathbf{s}_i^{h+1}|_\tau = (\alpha, D + D_i)|_\tau = \alpha$ and thus region $s_i \equiv \mathbf{s}_i^{h+1}|_\tau$ is the same as $s_i \equiv \alpha$.

Our goal now is to verify that $\Phi_{N,a} \vdash \Diamond_\tau(s_i \equiv \alpha)$. To this end, consider the state excitation hypothesis $s_i :=_{D_i} S_i$ in $\Phi_{N,a}$. If we could show that $\Phi_{N,a} \vdash \Diamond_D(S_i \equiv \alpha)$ then, as $\tau \geq D + D_i$, we also obtain $\Phi_{N,a} \vdash \Diamond_\tau(s_i \equiv \alpha)$ by rule $\supset\Diamond L$ and $\Diamond id$, as desired. So, why should $\Diamond_D(S_i \equiv \alpha)$ be derivable from $\Phi_{N,a}$? We remember that D, by assumption, is the minimal E such that $\mathbf{S}_i(\mathbf{x}^h|_E) = \alpha$, where $\mathbf{x}^h|_E = (val(a).\mathbf{s}^h)|_E = a.\mathbf{s}^h|_E$. In particular, then, $\mathbf{S}_i(\mathbf{x}^h|_D) = \mathbf{S}_i(a.\mathbf{s}^h|_D) = \alpha$.

Since $glb(S_i(set(\mathbf{x}^h|_D))) = \mathbf{S}_i(\mathbf{x}^h|_D) = \alpha \neq \bot$ we conclude $S_i(set(\mathbf{x}^h|_D)) = \{\alpha\}$. In other words, the evaluation of S_i in region $set(\mathbf{x}^h|_D)$ results in constant value α. But this must mean that $set(a.\mathbf{s}^h|_D) = set(\mathbf{x}^h|_D) \subseteq (S_i \equiv \alpha)$. It is finally time for the induction hypothesis $\Phi_{N,a} \vdash \Diamond_D set(a.\mathbf{s}^h|_D)$, which, together with rule $\Diamond id$, obtains $\Phi_{N,a} \vdash \Diamond_D(S_i \equiv \alpha)$. This is what we were after.

($\Leftarrow$) In the reverse direction let $\Phi_{N,a} \vdash \Diamond_\tau R$ be a derivation in UN-Logic for arbitrary region $R \subseteq \mathbb{B}^{\mathcal{Z}}$. We show by induction on its length that $set(a.[\![N]\!](a)|_\tau) \subseteq R$, viz. that there exists an iteration index $h \geq 0$ such that $set(a.\mathbf{s}^h|_\tau) \subseteq R$.

The base case is when no rule has been applied, i.e., $\Diamond_\tau R$ is contained in $\Phi_{N,a}$. Then, by construction of $\Phi_{N,a}$, this modal region $\Diamond_\tau R$ must be the input specification $\Diamond_0(\mathcal{X} \equiv a)$. Hence, $R = (\mathcal{X} \equiv a)$ and indeed $set(a.\mathbf{s}^0|_\tau) = set(a.\bot^m) = (\mathcal{X} \equiv a)$, irrespective of τ.

For the induction step we have four cases. First, $\Diamond_\tau R$ might be obtained by rule $\Diamond true$ which means that $R = 1$. But then trivially $set(a.\mathbf{s}^h|_\tau) \subseteq R$ for any $h \geq 0$, since R is the full space $\mathbb{B}^{m+n}$.

The second possibility is that $\Diamond_\tau R$ is constructed from some smaller derivation $\Phi_{N,a} \vdash \Diamond_D T$ by rule $\Diamond id$, i.e., $D \leq \tau$ and $T \subseteq R$. By induction hypothesis, there exists $h \geq 0$ with $set(a.\mathbf{s}^h|_D) \subseteq T$. Moreover, monotonicity implies $a.\mathbf{s}^h|_D \sqsubseteq a.\mathbf{s}^h|_\tau$ and hence $set(a.\mathbf{s}^h|_\tau) \subseteq set(a.\mathbf{s}^h|_D) \subseteq T \subseteq R$.

As the third case let us assume $\Diamond_\tau R$ is generated from a smaller derivation $\Phi_{N,a} \vdash \Diamond_D T$ applying rule $\supset\Diamond L$ to some implication $T \supset \Diamond_{D_i} R$ from $\Phi_{N,a}$. Then, $\tau = D + D_i$. Because of the way $\Phi_{N,a}$ is constructed, this implication $T \supset \Diamond_{D_i} R$ must be one half of a state excitation assignment $s_i :=_{D_i} S_i$, i.e., $T = (S_i \equiv \alpha)$ and $R = (s_i \equiv \alpha)$ for some $\alpha \in \mathbb{B}$. The induction hypothesis on $\Phi_{N,a} \vdash \Diamond_D T$ guarantees there exists $h \geq 0$ such that $set(a.\mathbf{s}^h|_D) \subseteq (S_i \equiv \alpha)$. But this means nothing but that excitation function S_i evaluates to α for each total state in $set(a.\mathbf{s}^h|_D)$, or $S_i(set(a.\mathbf{s}^h|_D)) = \{\alpha\}$, whence $\mathbf{S}_i(a.\mathbf{s}^h|_D) = glb(S_i(set(a.\mathbf{s}^h|_D))) =$

α. Moreover, this implies both $\mathbf{S}_i(a.\mathbf{s}^h|_\infty) = \alpha$ and $D' = min\{E \mid \mathbf{S}_i(a.\mathbf{s}^h|_E) \neq \bot\} \leq D$, so that $let_i(\mathbf{S})(a.\mathbf{s}^h) = (\alpha, D')$ and further $\mathbf{s}_i^{h+1} = let_i(\mathbf{S})(a.\mathbf{s}^h) + D_i = (\alpha, D' + D_i)$. Since $\tau = D + D_i \geq D' + D_i$ we finally find $\mathbf{s}_i^{h+1}|_\tau = \alpha$ and $set(a.\mathbf{s}^{h+1}|_\tau) = (\mathcal{X} \equiv a) \wedge \bigwedge_{j \leq m}(s_j \equiv \mathbf{s}_j^{h+1}|_\tau) \subseteq (s_i \equiv \mathbf{s}_i^{h+1}|_\tau) \subseteq (s_i \equiv \alpha) = R$. This was to be shown.

Finally, the forth and last case is a derivation of $\Diamond_\tau R$ from two smaller proofs $\Phi_{N,a} \vdash \Diamond_D T$ and $\Phi_{N,a} \vdash \Diamond_E S$ with the help of rule $\Diamond\wedge R$. Then, $R = T \cap S$ and $\tau = max(D, E)$. Because of the induction hypothesis we have indices $h_1 \geq 0$ and $h_2 \geq 0$ with $set(a.\mathbf{s}^{h_1}|_D) \subseteq T$ and $set(a.\mathbf{s}^{h_2}|_E) \subseteq S$. Let $h = max(h_1, h_2)$. Then, by monotonicity $\mathbf{s}^{h_1}|_D \sqsubseteq \mathbf{s}^h|_\tau$ and $\mathbf{s}^{h_2}|_E \sqsubseteq \mathbf{s}^h|_\tau$ which implies $set(\mathbf{s}^h|_\tau) \subseteq set(\mathbf{s}^{h_1}|_D) \cap set(\mathbf{s}^{h_2}|_E) \subseteq T \cap S = R$ as desired. This completes the induction proof of the reverse direction ($\Leftarrow$) for (8) and thus the proof of Theorem 5. □

We now bring together Soundness and Completeness Theorems 2 and 3 relating the real-time semantics $\models$ and logical deduction $\vdash$ on the one hand, with Theorem 5 that relates $\vdash$ and timed ternary simulation $[\![N]\!]$ on the other. The following corollary fills in the missing semantical justification for pure and timed ternary simulation [8, 23, 36] of cyclic circuits.

Corollary 3 (Algebraic characterization of constructive networks) *Timed ternary simulation $[\![N]\!]$ computes correct and exact UN-stabilization information, in both value and time, for every signal. In particular, N is constructive iff for all inputs a, and $i \leq m$ $[\![N]\!]_i(a) \neq \bot$.*

Malik's pure ternary simulation [23] or Brzozowski and Seger's Algorithm B [8] can be obtained from the model above by abstracting time. Instead of computing the fixed point $[\![N]\!](a)$ in $\mathbb{TS}^m$ we compute its final time projection $[\![N]\!](a)|_\infty$ in $\mathbb{S}^m$. First note that

$$\begin{aligned}
\mathbf{s}^{h+1}|_\infty &= \mathbf{N}\big(val(a).\mathbf{s}^h\big)|_\infty \\
&= \big(let(\mathbf{S})\big(val(a).\mathbf{s}^h\big) + \mathbf{D}\big)|_\infty \\
&= \big(let(\mathbf{S})\big(val(a).\mathbf{s}^h\big)\big)|_\infty \\
&= \mathbf{S}\big(\big(val(a).\mathbf{s}^h\big)|_\infty\big) \\
&= \mathbf{S}\big(a.\big(\mathbf{s}^h|_\infty\big)\big) \qquad (11)
\end{aligned}$$

for all $h \geq 0$. Hence, abstracting from the timing part, the value iteration behind (7) is $\mathbf{s}^{h+1}|_\infty = \mathbf{S}(a.(\mathbf{s}^h|_\infty))$. Thus, we obtain $[\![N]\!](a)|_\infty = \mu\mathbf{y}.\mathbf{S}(a.\mathbf{y})$, where $\mathbf{y}$ now is a fixed point variable ranging over $\mathbb{S}^m$. This is precisely the standard ternary simulation algorithm. Below we give Malik's ternary algorithm, adding back the internal combinational vertices. Soundness and exactness of the algorithm are direct consequences of Corollary 3.

Malik's Algorithm (explicit version)

```
1 h := 0;
2 s^0 := ⊥^m;
3 repeat
4     h := h + 1;
5     for each combinational vertex z_j ∈ Z \ (X ∪ S)
6         z_j^h := F_j(a.s^{h-1}); /* where F_j is the circuit function of z_j */
7     for each state variable s_i ∈ S
8         s_i^h := S_i(a.s^{h-1}); /* where S_i is the excitation function of s_i */
9 until s^h = s^{h-1};
```

Example 11 Let us apply the pure ternary simulation (without combinational vertices) to our example network N_1 from Sect. 2.1 with system specification is $\Phi_{N_1} = \{s_1 :=_{D_1} x + s_2, s_2 :=_{D_2} \overline{s_1} + \overline{x}{\cdot}s_2\}$. Its UN-delay behavior was described in Fig. 7. As we have seen, for input $x = 0$, N_1 exhibits oscillation under UN-delay assumptions and thus is not ternary combinational. In contrast, for inertial delays (UI-delay assumption) N_1 does stabilize (see Sect. 2.2.1 and Fig. 6). Let us have a closer look at the simulation process to see where the two models diverge.

For N_1 the state excitation function is $S\langle x, s_1, s_2\rangle = \langle x + s_2, \overline{s_1} + \overline{x}{\cdot}s_2\rangle$ whose ternary abstraction $\mathbf{S} : \mathbb{S}^3 \to \mathbb{S}^2$ of S is given by

$$\mathbf{S}(\mathbf{z}) = \big\langle glb\big(S_1\big(set(\mathbf{z})\big)\big), glb\big(S_2\big(set(\mathbf{z})\big)\big)\big\rangle.$$

To compute the fixed point response $[\![N]\!](0)|_\infty = \mu\mathbf{y}.\mathbf{S}(0.\mathbf{y})$ under input $x \equiv 0$ we iterate the mapping $\mathbf{y} \mapsto \mathbf{S}(0.\mathbf{y})$ starting from $\mathbf{s}^0 = \bot\,\bot$. As a state region this still is exact information under both UI and UN-delays: all of the four binary states $set(\mathbf{s}^0) = \{00, 01, 10, 11\}$ covered by $\mathbf{s}^0$ are possible as initial states. Applying the system excitation S to this region obtains $S(set(0.\mathbf{s}^0)) = \{S(0.t) \mid t \in set(\mathbf{s}^0)\} = \{S(0.00), S(0.11), S(0.10), S(0.01)\} = \{00, 01, 11\}$. Notice that this is a *contraction* in the sense that $S(set(0.\mathbf{s}^0)) \subsetneq set(\mathbf{s}^0)$. The state 10 is no longer in the excitation set. For UI-delays this would imply that no state change out of the unknown initial state $\mathbf{s}^0$ can result in 10. Continuing over *arbitrary* regions, we would find stabilization. Indeed, $S(0.\{00, 01, 11\}) = \{01, 11\}$, and finally $S(0.\{01, 11\}) = \{11\}$, which is a fixed point. Note that in this approximation sequence all regions except $\mathbf{s}^1$ are expressible by ternary vectors. The fact that $\mathbf{s}^1$ is not makes the UN-delay model, which operates under ternary values, differ from UI-delays. The tightest ternary vector to approximate the region $\{00, 01, 11\}$ is $\bot\,\bot$. But this is precisely what determines the next iteration of the ternary extension $\mathbf{S}$, viz. $\mathbf{s}^1 = \mathbf{S}(0.\mathbf{s}^0) = glb(S(set(0.\mathbf{s}^0))) = glb\{00, 01, 11\} = \bot\,\bot$. So, nothing is gained over $\mathbf{s}^0$ and $[\![N]\!](0) = \mu\mathbf{y}.\mathbf{S}(0.\mathbf{y}) = \bot\bot$ is the fixed point and final result.

As presented, the explicit ternary algorithm would have to be executed 2^n times, once for each input combination, to compute $[\![N]\!](a)|_\infty$ for all a. In fact, the algorithm proposed by Malik [23] works on symbolic input values, using BDDs. In effect, all 2^n cases are handled in parallel, with possible sharing of work among the cases. It constructs a symbolic representation of all input combinations under which the circuit is UN-combinational.

The conversion from the explicit to the symbolic algorithm is straightforward. A ternary valued vertex function, which is updated on each iteration, is stored at each vertex. These equations are defined over the circuit inputs. Since the inputs are assumed to be binary valued, the functions to be represented are of the form $\mathbf{F} : \mathbb{B}^n \to \mathbb{S}$. Such functions are in turn represented by a pair of Boolean functions F^1 and F^0, where F^1 (resp. F^0) is the characteristic function of the set of inputs for which $\mathbf{F}$ evaluates to 1 (resp. 0). The set of inputs for which $\mathbf{F}$ evaluates to $\bot$ is computed as $F^\bot = \overline{F^1 + F^0}$. The functions F^1 and F^0 are coded as BDDs.

To start the algorithm, each input is initialized to a Boolean symbolic variable, and each vertex function corresponding to a state variable is initialized to the constant function $\bot$. As before, within each round, the gates are visited in topological order. For each gate, the new vertex function is computed by combining the vertex functions of lower depth according to the Boolean operation implied by the gate's vertex function V. For example, if V is the Boolean conjunction of two vertices represented by the functions $\mathbf{G}$ and $\mathbf{H}$, then the new vertex function for $\mathbf{F}$ is given by $F^1 = G^1{\cdot}H^1$ and $F^0 = G^0 + H^0$. The algorithm repeats

until none of the vertex functions at the state variables change from one round to the next. Convergence is guaranteed within m rounds, where m is the number of state variables. Correctness and exactness of the symbolic algorithm follow from Corollary 3 and the fact that it is a symbolic implementation of the explicit algorithm. Namjoshi and Kurshan [31] have presented an improvement, based on SAT-solving, which is more efficient and also generalizes the symbolic analysis from Boolean to arbitrary value domains. Other SAT-based approaches for cyclic circuits are [1, 34]. Riedel and Bruck [36] show how the symbolic ternary algorithm can be extended to include timing. However, none of these works takes account of real-time behavior as we do here.

4 Conclusion

In the presence of cycles, the notion of combinational circuits is not unique but depends on the chosen delay model and operating conditions. Any design methodology for cyclic combinational circuits must make these choices clear in order to be electrically meaningful. In this article we study the mathematical underpinning for the class of constructive circuits, which has been treated only informally in the literature so far. In this section we sum up our contributions in the following areas:

1. Non-inertial delay model;
2. Constructive circuits and timed ternary simulation;
3. Circuit specification language.

4.1 Non-inertial delay model

Constructive networks coincide with the class of UN-combinational networks, where UN is the conservative non-inertial up-bounded delay. We have argued that the UN-delay model should replace the traditional UI-delay in the construction and analysis of combinational systems. The non-inertial UN-delay is related to the "XBD0" delay considered by McGeer *et al.* [25] for floating mode path sensitization on non-cyclic circuits and the "binary chaos" delay model of Burch [2] used in the analysis of speed-independent asynchronous circuits.

The salient feature of the UN-model highlighted in this article is its semantical constructivity which means that all anomalous behavior (specifically non-determinism and metastability) can be treated as oscillation. This makes network analysis simple and efficient as it can be performed in the ternary domain $\mathbb{S} = \{0, 1, \bot\}$ in an abstract fashion without losing information in compositional reasoning. An evaluation $z = \bot$ for the stationary behavior of a network vertex associates with z, at the same time, a stable but unknown Boolean value (non-determinism) and a canonical unstable but known behavior (oscillation). Both non-determinism and oscillation may occur in real circuits but are not the only way of undecidedness (see [48]). It is also possible that metastability is observed as a signal lingering in an intermediate value for unbounded time (see [17, 24]). The *binary-valued* histories used in this article cannot represent this. However, we believe that our completeness and constructiveness results can be extended to continuous-valued histories. This would permit us to rephrase Corollary 2 so it implies the existence of a single 'infinitely undecided' history in addition to 'oscillation', in case of non-determinism or unboundedness. This might then lead to a physically more realistic gate delay model in the sense of Stephan and Brayton [44].

In principle, it should be possible to ground delay models in a mathematical model of the solid-state physics of the fabricated circuits, including perhaps a variety of low-level parameters capturing the process variations of silicon technology (e.g., determining wire loads

and parasitic capacitances) and concrete physical operating conditions (e.g., temperature and supply voltage). A formal relationship between the UN-delay and an electrical model of silicon circuits would require a theory of behavioral refinement (cf. the notion of "behavior epimorphism" of Stephan and Brayton). This is beyond the scope of this article and left as an open question.

4.2 Constructive circuits and timed ternary simulation

The UN-model comes with a coherent mathematical theory. As we have shown, UN-combinational networks can be characterized in three different ways: The *electrical definition* corresponds to termination and functionality with respect to the UN-delay model. The *logical definition* is given in terms of a simple constructive modal logic, called UN-Logic. A network N is constructive if for every signal s a formula $\Diamond_D s$ or $\Diamond_D \overline{s}$ can be proven from the N's logical specification (for all input assignments). The *algebraic definition* uses Malik's algorithm of ternary simulation, for which a direct electrical interpretation has been missing in the literature. It is shown here that for UN-delays the electrical, logical and the algebraic definitions coincide on all networks. The electrical and algebraic views have direct practical relevance, and the logical definition in terms of UN-Logic is a useful conceptual bonus: It provides a natural refinement of Boolean logic to bridge the semantic gap between the electrical definition of UN-delays and ternary simulation.

This work extends the results by Shiple et al. [37, 41], where the coincidence between ternary simulation and the electrical definition was shown in the special case of complete networks and UI-delays. As we have pointed out, that result does not hold up for arbitrary UI-networks. To remedy this, we offer a new definition based on the UN-delay model. Adopting the UN-model, which is strictly more conservative, allows us to leverage the power of ternary simulation to simplify the analysis of *arbitrary* networks rather than just complete ones. As discussed below, on complete networks the classes of UI-combinational and UN-combinational networks coincide. Still, on complete networks we get a stronger result: Knowing that a complete network is combinational under the UN-delay model is electrically more informative and reassuring than knowing it is combinational for UI-delays, because the former covers more behavior and is electrically more realistic than the latter. For networks classified as 'constructive' we are assured that there is no anomalous behavior (oscillation, non-determinism or metastability) at the electrical level under a delay model more conservative than most other delay models considered in the literature so far.

This work has practical applications because many traditional design automation tools, such as cycle-based simulators and formal verification tools, do not accept circuits with combinational cycles, or they handle them in an ad-hoc manner. Yet, such circuits are attractive for hardware synthesis because they can be more area and time efficient compared to equivalent acyclic circuits, as shown by Riedel and Bruck [35, 36]. Our theory can be used to extend tools for cyclic combinational circuits on a firm mathematical basis. Specifically, our presentation of symbolic simulation in UN-Logic which includes upper-bound timing provides a mathematical underpinning of Riedel and Bruck's algorithm [36]. We show that their algorithm not only computes correct and exact settling times under the UN-delay model but that it can be generalized to handle arbitrary real-valued delays and arbitrary non-complete circuits.

4.3 Circuit specification language

A main technical contribution of this work, besides proving the equivalence between the ternary simulation model and UN-steady-state behavior is the introduction of UN-Logic

and the application of powerful techniques from logic and model theory to bridge different computational domains. Since we interpret formulas on time intervals, UN-Logic is an interval temporal logic. The modality $\Diamond_D$ is reminiscent of an eventuality operator of temporal logics. Formally, $\Diamond_D$ is known as a *lax modality* capturing a weakened notion of truth "up to some constraints" satisfying the characteristic axioms

$$\phi \supset \Diamond_D \phi$$
$$\Diamond_D \Diamond_E \phi \supset \Diamond_{D+E} \phi$$
$$\Diamond_D \phi \wedge \Diamond_E \psi \supset \Diamond_{max(D,E)}(\phi \wedge \psi)$$

and the rule $\models \phi \supset \psi \;\Rightarrow \models \Diamond_D \phi \supset \Diamond_D \psi$. In type theory and functional programming such modalities are called *strong monads* [28]. They are typically used to capture intensional features such as side effects, non-determinism, partiality, and other notions of computations. The application of lax modalities for constraints has been proposed by Fairtlough and Mendler [14] and its application to timing was investigated in [26, 27]. In [27] the fragment of UN-Logic consisting of Horn clauses $R \supset \Diamond_E S$, where R and S are *cubical regions*, has been shown to give correct and exact timing information with respect to a discrete time semantics. The Soundness and Completeness Theorems 2 and 3 presented in this article generalize these results to continuous time and arbitrary regions.

In this article, we have not used the full potential of UN-logic, which is a constructive refinement of both Boolean algebra and ternary algebra:[9]

- It turns out that all modalities $\Diamond_D \phi$, for $D > 0$ trivialize to *true* under double negation, and disjunction $\phi \vee \psi$ becomes set union. In this way, doubly-negated formulas are simply regions, and the double negation fragment of the logic can be identified with classical Boolean algebra.
- The UN-calculus presented in Sect. 3.2 is tailored to network theories. This covers the syntax corresponding to ternary algebra, viz. the modal Horn-clause fragment in which all regions are conjunctions of atomic equations $z \equiv 0$, $z \equiv 1$ or, which amounts to the same, conjunctions of literals z, $\overline{z}$. It is relative to this fragment that we have proved the computational adequacy of timed ternary simulation.
- Using arbitrary regions as the atomic propositions takes us beyond ternary algebra. For instance, $(y_1 \equiv y_2) :=_D x$ would model a gate with one input x and two outputs y_1, y_2 forcing both outputs become equal if x is high and different if x is low, without specifying which values the outputs take. This even permits oscillations within each of the non-cubical regions $y_1 \equiv y_2$ and $y_1 \not\equiv y_2$, respectively. Such non-deterministic response cannot be modeled by a ternary function, because it can only handle cubical regions. Similarly, $y :=_D (x_1 \equiv x_2)$ would model a component with output y driven high by two input nodes x_1, x_2 being identical for a sufficient amount of time. This cannot be expressed by any ternary function.

Mendler [26] gives an idea of the surprising expressiveness of constructive modal logic regarding stabilization behavior and data-dependent delays. We leave the construction of a full-fledged UN-calculus and the characterization of its expressiveness as an open problem.

[9]Ternary *algebra* itself is sometimes interpreted as a *logic* of three truth-values in analogy to propositional logic which is the logic of two truth values. This analogy is misleading, however, as ternary algebra is *not* a logic. The reason is that it does not have any theorems, i.e., non-trivial ternary expressions that evaluate to "truth value" 1 under *all* ternary valuations.

The advantage of the axiomatic approach advocated here is that it can be applied to other kinds of delay models, too. Specifically, the UI-delay model used by Brzozowski and Seger can be specified by way of a *necessity* modality in an extension of our logic. A suitable modal operator $\Box\phi$ might be defined as follows:

$$h[s,t) \models \Box\phi \quad \text{iff } s \geq t \quad \text{or} \quad \exists \epsilon > 0.\, h[s, t+\epsilon) \models \phi.$$

Thus, $\Box\phi$ holds in a non-empty interval $[s,t)$ if ϕ is true in $[s,t)$ but also extends positively beyond t. This can be interpreted as a form of *inertial truth*. Stability (or memory, for that matter) can now be expressed easily assuming non-Zeno and right-continuous histories. The formula $x \supset \Box x$, for instance, says that whenever x becomes 1 for any positive interval of time, then it latches and remains 1 forever. A UI-delay between S and s can be captured by the formula

$$\mathrm{UI}_D(S,s) =_{df} (S \wedge s) \supset \Box s \ \wedge \ (\overline{S} \wedge \overline{s}) \supset \Box \overline{s} \ \wedge \ s :=_D S.$$

A sound and complete formal calculus for both $\Diamond$ and $\Box$ might then yield a precise characterization of the expressiveness of Brzozowski and Seger's ternary simulation algorithms A and B, essentially filling the gap left by their completeness theorems [8] [Theorems 7.1, 7.2] which apply to complete networks only.

5 Related work

We divide related work into two categories.

1. Circuit analysis: These works provide techniques to analyze the behavior of circuits, without attempting to classify well-behaved circuits.
2. Circuit classification: These works classify circuits based on their well-behavedness.

5.1 Circuit analysis

There is a large body of literature on the timing analysis of gate-level circuits, most of which is restricted to acyclic networks and also does not prove exactness relative to an electrical delay model. Typically, timing analyses aim to compute the longest *sensitizable path* from input to output. Exactness issues, if treated at all, are then expressed relative to the chosen sensitization criterion. See, e.g. the paper by Marques Silva and Sakallah [42] for an overview. Among the few works which formally ground timing analyses in electrical delay models are the publications by McGeer et al. [25], Lam and Brayton [21], and Mendler [26] but these do not deal with cycles.

Research on cyclic circuits traditionally is done in the context of asynchronous design methodologies. We refer to the work of Unger [47, 48] and Brzozowski and Seger [8] for a survey of original work, specifically, on ternary hazard analysis for asynchronous circuits. Work in this area differs from ours in two ways. Firstly, asynchronous circuit analysis aims to analyze the *transient* and *sequential* behavior, rather than *combinational stabilization* properties addressed here. Delay elements for asynchronous systems must have inertiality properties, i.e., memory, in order to explain sequential state transitions. For contrast, constructive circuits used in synchronous designs ensure that the outputs are uniquely and functionally driven by the inputs (and the value of registers) at all times. This excludes any form of memory. Secondly, traditional analyses for asynchronous circuit dynamics enrich the Boolean

signal domain by additional non-standard values to capture timing- or signal-related low-level electrical phenomena. See, e.g., Brzozowski et al. [7] for a survey of multi-valued hazard algebras and their applications. This differs from the axiomatic approach advocated here. Many-valued "signal" domains, from our point of view, should not be seen as concrete representations of signal features manifest in physical circuits. Instead, as we have shown, the ternary algebra arises as a convenient and computationally adequate domain to decide a particular logic theory. It is the logical theory that captures low-level physical behavior not the ternary values. We believe that our approach generalizes to derive exactness results for other many-valued hazard algebras in a similar way.

Maler and Pnueli [29] provide an elegant method to translate asynchronous circuits, described at the gate-level, into timed automata. They use a delay model equivalent to the bi-bounded inertial delay of Brzozowski and Seger; this is more general than the up-bounded inertial model because it allows the specification of a lower bound on the delay. For each gate in the circuit, they introduce a delay element with an associated timer (or clock). They prove that the resulting timed automaton has the same I/O behavior over time as the original circuit, for the given delay model. Using the timed automaton, they are able to perform state reachability and solve several synthesis problems. However, they do not address the well-behavedness problem, nor is it immediate how this problem can be solved within their framework. The use of timers complicates the analysis considerably. This is unnecessary for combinational and clocked sequential systems where UN-delays can be used, since there constructivity is independent of the delay bounds as we show.

Brzozowski and Seger [8] study general asynchronous circuits under various delay models, specifically the up-bounded inertial delay and the bi-bounded inertial delay. They present algorithms for both models to analyze the sequential behavior of cyclic circuits. It is shown that the sequential behavior of complete UI-delay networks can be decided by a two-pass ternary simulation, called *Algorithm A* and *Algorithm B*. In this work, the inertiality property is crucial to explain memory effects. Brzozowski and Seger do not consider non-inertial delays or the classification of combinational (i.e., non-sequential) circuits. However, their results imply that Algorithm B decides the class of UI-combinational complete networks. This has been shown independently by Shiple [41]. In the present article we derive the stronger result that Algorithm B (used for combinational analysis) is correct and exact for *arbitrary* networks under the *weaker* UN-delay model. Our results (Sect. 2.2) also imply that Algorithm B is incomplete for non-complete networks.

As far as we are aware, Malik [23] is the first to use ternary simulation, corresponding to Algorithm B for *cyclic* and *combinational* systems. Later, Riedel and Bruck [36] extended Malik's symbolic ternary simulation to quantitative timing. None of these considers exactness issues. Fairtlough and Mendler [27] are the first to discuss the exact timing for cyclic combinational systems based on the non-inertial delay model, and also to investigate the connection with ternary simulation.

Burch et al. [3] model logic gates by ternary-valued relations, where the third value, $\perp$, represents an oscillating or intermediate voltage. By using $\perp$, oscillating behaviors caused by combinational cycles are preserved when gates are composed (by taking the intersection of their corresponding ternary-valued relations). This is in contrast to the use of Boolean relations to model gates, where oscillating behaviors "disappear" when gates are composed. Burch uses the ternary model to solve various substitution and rectification problems for gate-level circuits. However, they do not classify well-behaved circuits, or relate the ternary model to a delay model.

Shiple, Berry and Touati [40] extend the definition of constructivity to clocked sequential circuits with flip-flops. Beyond the obvious relevance to hardware design, this extension

plays a critical role in compiling programs in Esterel, a language for embedded software. In particular, Berry [4] shows that an Esterel program is well-behaved if and only if the circuit derived from the program is constructive. These are known as constructive Esterel programs. The algorithm for deciding clocked constructive circuits consists of an initial call to Malik's algorithm, followed by implicit state enumeration over the reachable state space of the flip-flops. This distinguishes between behavior in the reachable and unreachable state space, but does not permit combinational wires to hold state. However, due to this last restriction, the inputs can have glitches, and consequently, this class is closed under cascade composition. Motivated by applications in synchronous programming, the restriction has been lifted by Schneider et al. [38].

Namjoshi and Kurshan [31] have shown that the algebraic definition of constructivity (Corollary 3) can be rephrased as a satisfiability problem. They observed that the total definedness of the least fixed-point $[\![N]\!](a) = \mu \mathbf{y}.\mathbf{S}(a.\mathbf{y})$ is equivalent to the existence of partial state values $\mathbf{s} \in \mathbb{S}^m \setminus \mathbb{B}^m$ such that $\mathbf{s} = \mathbf{S}(a.\mathbf{s})$. The same definition has also been used later in the work of Claessen [10] and Backes et al. [1] on the analysis of cyclic circuits. Like Namjoshi and Kurshan these works do not capture quantitative timing nor relate their analyses with an electrical delay model or a logical definition, as we do here. As reported in [31] the SAT-solving technique can be more efficient than Malik's symbolic algorithm, and also generalizes by replacing $\mathbb{B}$ with an arbitrary value set T, and $\mathbb{S}$ by its canonically lifted Scott domain $T_\perp$. However, this depends on efficient representations for the lifting $\mathbf{S}$ of the network's excitation function S. In [1, 10, 31] the lifting $\mathbf{S}$ is obtained by composing the liftings of the basic Boolean operations from which S is constructed. This is equivalent to analyzing the constructiveness of *complete* networks. For non-complete networks, considered here, the operations in S must be composed first and then lifted to $\mathbf{S}$. While this is still reasonable for Boolean functions, it may be impractical on arithmetical value domains suggested in [31]. Also, it is not obvious how timing analysis might be added to the SAT-solving approach.

5.2 Circuit classification

The work of Malik [23] and Brzozowski and Seger [8] inspired our research. Malik notes that combinational cycles do arise in practice, but that no method for rigorously analyzing such circuits had ever been proposed. To remedy this situation, he introduced the class of "combinational" circuits to capture well-behavedness, and proposed using ternary simulation to decide whether or not a circuit is combinational. However, his work is not based on a delay model.

Shiple, Berry and Touati [40] introduced the term "constructive" circuit in line with the terminology adopted in the synchronous language community. They sketched the three alternative characterizations, the logical, algebraic and electrical definition. The main objective of the present article is to give precise mathematical definitions to these interpretations and to provide a formal proof of their relationships which so far has only been informally discussed in the literature. Specifically, we have shown that constructive networks are the same as Malik's ternary combinational networks and both co-extensional with UN-combinational networks.

This article aims to understand combinational systems in terms of semantical criteria. In the following we will briefly indicate the options arising from this more systematic vantage point. Specifically, we take account not only of the delay model but also on the placement of the delay elements and the operating conditions.

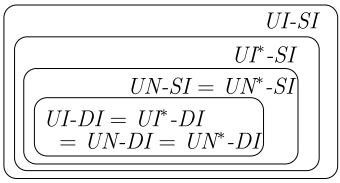


Fig. 14 Strict hierarchy of circuit classes

Take a Boolean circuit C and vary the number and placement of delay elements that make up the network N associated with C. Every additional delay desynchronizes the computations and makes oscillations more likely. For instance, while a singleton UI-delay loop is still terminating, two UI-delays in a cycle can oscillate under simultaneous switching of both elements (Fig. 13 on the right), three UI-delays may even oscillate by changing only a single bit at a time. The UN-delay, in contrast, shows oscillation already for an atomic loop. Obviously, circuit analysis is more conservative the more delays we use to form the network and the more transient non-determinism is exhibited by the delay elements.

In [30, Fig. 16, Sect. 4] a circuit is presented which is combinational for some distributions of UN-delays but not for others, while in the latter cases it can become combinational by replacing the UN-delays by stronger UI-delays. In particular, this shows that UN-combinational networks are a *proper* sub-class of the UI-combinational networks and it highlights the non-compositional nature of UI-delays. Increasing the number of UI-delays at the output of a gate can cost stabilization. An arbitrary sequence of UN-delays, in contrast, is always equivalent to a single UN-delay (modulo adjustment of the delay parameter). UI-delays are not closed under iteration in this way. Intuitively, the reason is that a sequence of l UI-delays can generate l successive signal transitions at the final output before it must stabilize (without input changes), while a single UI-delay can only buffer one signal transition.

In general, variations in multiplicity and placement give rise to notions such as *delay-insensitivity* and *speed-independence*. A circuit C is *DEL-speed-independent* (*DEL-SI*) if all gate-delay networks obtained from C are DEL-combinational and it is *DEL-delay-insensitive* (*DEL-DI*) if all gate, input and wire-delay networks of C are DEL-combinational. In each case it is understood that the multiplicity of delays is 1. We can strengthen the notions to say that C is *DEL***-SI* or *DEL***-DI*, respectively, if C is *DEL-SI* or *DEL-DI* for arbitrary multiplicity of delays. It is clear from the definition that *DEL***-DI* $\subseteq$ *DEL-DI* $\subseteq$ *DEL-SI* and *DEL***-DI* $\subseteq$ *DEL***-SI* $\subseteq$ *DEL-SI*. As far as delay-insensitivity is concerned, our results imply that there is no difference between the delay-models, i.e., *UN-DI* $=$ *UI-DI* and *UN***-DI* $=$ *UI***-DI*. Why? Observe that a gate, input and wire-delay model of a circuit, irrespective of multiplicities, is a *complete* network which can be decided by ternary simulation [8, 37] for UI-delays. By Theorem 5 ternary simulation also decides stabilization under the UN-delay model for arbitrary networks. Hence, a circuit is *UI-DI* (*UI***-DI*) if and only if it is *UN-DI* (*UN***-DI*). Moreover, as noted above, we have *UN-DI* $=$ *UN***-DI* for the UN-delay model, whence all four delay insensitive classes *UN-DI*, *UN***-DI*, *UI-DI*, *UI***-DI* coincide. Regarding speed-independence the situation is more interesting. Figure 14 sums up our findings (see [30, Sect. 4]). We obtain a strict hierarchy of four circuit classes from the eight possible combinations, varying between the UI and UN-delay model, speed-independence and delay-insensitivity, single and arbitrary multiplicities.

Besides the UI and UN-delay models there are many other delay-models to be investigated, e.g., the *bi-bounded* (inertial and non-inertial) delays introduced in [8] and the *pure* delay as it is used in the work on Timed Boolean Functions [21]. Regarding the selection of state variables one traditionally considers the *gate-state* network model and the *gate, input*

and wire-state network, known as the complete network model. Among the *feedback-state* networks these constitute only two out of many other possible choices. Specifically, delay multiplicities have an important role to play in characterizing a circuit's robustness with respect to internal buffering.

Finally, there are variations in the operating conditions. For combinational systems it is natural to assume, as we have done in this work, that all inputs are held constant. In other words, the input excitation in every execution of C is described by a single input vector, which may be constrained in terms of *don't care* sets. Instead of one input vector one could apply sequences of input vectors, obtaining what is called the *n-vector* delay [21]. The other element in fixing the operating condition relates to the initial state, which is determined by the behavior of the input before time 0. We may let C start in a *stable* or an *arbitrary* initial state. A stable initial state would correspond to a constant input in the interval $(-\infty, 0)$ which is switched to a new constant value at time 0. In standard terminology [21] this is called the *2-vector mode*. Working under the arbitrary initial state assumption, which is what we do here, is known as *floating mode*. Also, (see e.g., [8]) there is a difference between *fundamental mode* where we require all signals to stabilize and *input-output mode* where stabilization is with respect to a limited set of observable output signals.

While a systematic classification of combinational systems has considerable theoretical interest, its relevance for conventional design practice is arguable. Exploiting such results practically, it seems, would require dedicated design styles that control the implementation of delay models and operating conditions reliably. For speed-independent circuits specialized design techniques exist, see e.g. [18], applicable to general asynchronous rather than combinational systems. The most conservative stand regarding cyclic combinational systems appears to be the class of *constructive circuits* investigated in this article. These are characterized by (1) fundamental mode stabilization, (2) floating mode initialization, (3) complete network model with arbitrary multiplicities, and (4) up-bounded non-inertial delays. This is precisely the class $UN^*\text{-}DI = UN\text{-}DI = UI^*\text{-}DI = UI\text{-}DI$ at the bottom of the hierarchy in Fig. 14. The fact that constructive circuits coincide with the delay-insensitive circuits under both the UN- and UI-delay model is a remarkable result. It means that constructive circuits are not only well-behaved under the UI-model, which follows from earlier results [37, 41]. Here we show that, even under the UN-delay model, which is more conservative than most delay-models considered in the literature in this context, we are assured that there is no anomalous behavior for circuits classified as 'constructive'.

Halbwachs and Maraninchi [16] define a class of well-behaved circuits called *consistent* circuits. Basically, they view a circuit as a system of Boolean equations (one equation for each gate), and consider the solutions of this system. For a given input valuation, if the system has at least one solution, and the output has the same value for all solutions, then the circuit is deemed *weakly consistent*. As a special case, if there is exactly one solution, then the circuit is *strongly consistent*. A similar approach has been taken by Schneider, Brandt and Schuele [38] for circuits with flipflops. This class is not comparable to our class of constructive circuits. There are circuits [23] [Fig. 6a] which are strongly consistent, but not constructive. On the other hand, every constructive circuit is strongly consistent because if there is more than one Boolean solution to the circuit equations, then (under both UI- and UN-delays) the circuit can oscillate consistently between them (this follows from our Constructiveness Theorem). If we run circuits in input-output mode, i.e., require constructivity only for a fixed set of observable outputs rather than all circuits vertices, the two notions diverge further: There are circuits which are constructive relative to some output [23] [Fig. 4b, for output z] but not even weakly consistent. The notion of strong consistency, or *logical correctness* [4], can be used to define a class of combinational systems, though its significance for circuit design (as opposed to distributed software) is unclear.

Acknowledgements We are grateful to the anonymous reviewers for their suggestions to improve this article. The first author was supported by the European Community as a member of the TYPES Project FP6-IST-510996 and the German Research Foundation DFG through the project grant "Precision-timed Synchronous Processing (PRETSY)".

References

1. Backes J, Fett B, Riedel M (2008) The analysis of cyclic circuits with Boolean satisfiability. In: Proc int'l conf on computer-aided design (ICCAD'08), pp 143–148
2. Burch JR (1992) Delay models for verifying speed-independent asynchronous circuits. In: Proc int'l conf computer design (ICCD'92), pp 270–274
3. Burch JR, Dill D, Wolf E, De Micheli G (1993) Modeling hierarchical combinational circuits. In: Proc int'l conf on computer-aided design, November 1993, pp 612–617
4. Berry G (1999) The constructive semantics of Esterel. Draft, version 3.0, available at www.esterel.org, July 1999
5. Breuer MA (1972) A note on three-valued logic simulation. IEEE Trans Comput C-21(4):399–402
6. Bryant RE (1987) Boolean analysis of MOS circuits. IEEE Trans Comput-Aided 6(4):634–649
7. Brzozowski JA, Ésik Z, Iland Y (2001) Algebras for hazard detection. In: Proc symposium on multiple-valued logic (ISMVL'01), pp 3–12
8. Brzozowski JA, Seger C-JH (1995) Asynchronous circuits. Springer, New York
9. Brzozowski JA, Yoeli M (1979) On a ternary model of gate networks. IEEE Trans Comput C-28:178–184
10. Claessen K (2004) Safety property verification of cyclic synchronous circuits. In: Synchronous languages, applications, and programming SLAP 2003. ENTCS, vol 88. Elsevier, Amsterdam, pp 55–69
11. Davey BA, Priestley HA (2002) Introduction to lattices and order. Cambridge University Press, Cambridge
12. de Simone R (1996) Note: a small hardware bus arbiter specification leading naturally to correct cyclic description. Internal note: http://www-sop.inria.fr/meije/verification/esterel/doc.html
13. Eichelberger EB (1965) Hazard detection in combinational and sequential switching circuits. IBM J Res Dev 9(2):90–99
14. Fairtlough F, Mendler M (1997) Propositional lax logic. Inf Comput 137(1):1–33
15. Gordon MJC (1979) The denotational description of programming languages. Springer, New York
16. Halbwachs N, Maraninchi F (1995) On the symbolic analysis of combinational loops in circuits and synchronous programs. In: Euromicro'95, September 1995, Como, Italy
17. Kinniment DJ (2007) Synchronization and arbitration in digital systems. Wiley, New York
18. Kishinevski M, Kondratyev A, Taubin A, Varshavsky V (1994) Concurrent hardware: the theory and practice of self-timed design. Wiley, New York
19. Kleene SC (1952) Introduction to metamathematics. North Holland, Amsterdam. Chap XII, Par 64
20. Lamport L (2003) Arbitration-free synchronization. Distrib Comput 16(2–3):219–237
21. Lam KC, Brayton RK (1994) Timed boolean functions. A unified formalism for exact timing analysis. Kluwer Academic, Norwell
22. Lloyd JW (1984) Foundations of logic programming. Springer, Berlin
23. Malik Sharad (1994) Analysis of cyclic combinational circuits. IEEE Trans Computer-Aided Des 13(7):950–956
24. Marino LR (1981) General theory of metastable operation. IEEE Trans Comput 30(2):107–115
25. Mc Geer P, Saldanha A, Brayton R, Sangiovanni-Vincentelli A (1992) Delay models and exact timing analysis. In: Sasao T (ed) New directions in logic synthesis and optimization. Kluwer, Norwell, pp 167–190
26. Mendler M (2000) Characterising combinational timing analyses in intuitionistic modal logic. Log J IGPL 8(6):821–853. Abstract appeared *ibid.* Vol 6, No 6 (Nov 1998)
27. Mendler M, Fairtlough F (1996) Ternary simulation: A refinement of binary functions or an abstraction of real-time behaviour. In: Sheeran M, Singh S (eds) Proceedings of the 3rd workshop on designing correct circuits (DCC96), October 1996. Springer, Berlin. Springer Electronic Workshops in Computing
28. Moggi E (1991) Notions of computation and monads. Inf Comput 93:55–92
29. Maler O, Pnueli A (1995) Timing analysis of asynchronous circuits using timed automata. In: Camurati PE, Eveking H (eds) Proceedings of the conference on correct hardware design and verification methods, Frankfurt/Main, Germany, October 1995. LNCS, vol 987, Springer, Berlin pp 189–205
30. Mendler M, Shiple T, Berry G (2006) Constructive boolean circuits and the exactness of ternary simulation. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik, vol 68. University of Bamberg, August 2006

31. Namjoshi KS, Kurshan RP (1999) Efficient analysis of cyclic definitions. In: CAV 1999. LNCS, vol 1633, pp 394–405
32. Pěchouček M (1976) Anomalous response times of input synchronizers. IEEE Trans Comput 25(2):133–139
33. Plotkin GD (1977) LCF as a programming language. Theor Comput Sci 5(3):223–256
34. Riedel M, Bruck J (2003) Cyclic combinational circuits: Analysis for synthesis. In: Int'l workshop on logic synthesis
35. Riedel MD, Bruck J (2003) The synthesis of cyclic combinational circuits. In: DAC, June 2003. ACM, New York
36. Riedel MD, Bruck J (2004) Timing analysis of cyclic combinational circuits. In: Int'l workshop on logic and synthesis, Temecula Creek, CA
37. Shiple TR, Brayton RK, Berry G, Sangiovanni-Vincentelli AL (2002) Logical analysis of combinational cycles. Technical Report UCB/ERL M02/21, EECS Department, University of California, Berkeley. This is a revision of selected parts of Shiple's PhD thesis [41]
38. Schneider K, Brandt J, Schuele T (2004) Causality analysis of synchronous programs with delayed actions. In: Conference on compilers, architecture, and synthesis for embedded systems, (CASES), Washington DC, USA, September 2004. ACM, New York, pp 179–189
39. Schneider K, Brandt J, Schuele T, Tuerk T (2005) Maximal causality analysis. In: Conference on application of concurrency to system design (ACSD), St Malo, France, June 2005. IEEE Comput Soc, Los Alamitos, pp 106–115
40. Shiple TR, Berry G, Touati H (1996) Constructive analysis of cyclic circuits. In: Proc European design and test conference, March 1996, pp 328–333
41. Shiple TR (1996) Formal analysis of synchronous circuits. PhD thesis, UC Berkeley, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, October 1996. Memorandum No. UCB/ERL M96/76
42. Marques Silva JPM, Sakallah KA (1993) An analysis of path sensitization criteria. In: Proc ICCD'93, pp 68–72
43. Srinivasan A, Malik S (1996) Practical analysis of cyclic combinational circuits. In: IEEE custom integrated circuits conference, pp 381–384
44. Stephan PR, Brayton RK (1993) Physically realizable gate models. In: Proc ICCD'93, pp 442–445
45. Stok Leon (1992) False loops through resource sharing. In: Proc int'l conf on computer-aided design, November 1992, pp 345–348
46. Tarski A (1955) A lattice-theoretical fixedpoint theorem and its applications. Pac J Math 5:285–309
47. Unger SH (1969) Asynchronous sequential switching circuits. Wiley Interscience, New York
48. Unger SH (1995) Hazards, critical races, and metastability. IEEE Trans Comput 44(6):754–768
49. Watanabe Y, Brayton RK (1993) The maximum set of permissible behaviors for FSM networks. In: Proc int'l conf on computer-aided design, November 1993, pp 316–320
50. Yoeli M, Brzozowski JA (1977) Ternary simulation of binary gate networks. In: Dunn JM, Epstein G (eds) Modern uses of multiple-valued logic. Reidel, Dordrecht, pp 41–50
51. Yoeli M, Rinon S (1964) Application of ternary algebra to the study of static hazards. J ACM 11:84–97