Characterising Combinational Timing Analyses in Intuitionistic Modal Logic

Abstract

The paper presents a new logical specification language, called Propositional Stabilisation Theory (PST), to capture the *stabilisation behaviour* of combinational input-output systems. PST is an intuitionistic propositional modal logic interpreted over sets of waveforms. The language is more economic than conventional specification formalisms such as timed Boolean functions, temporal logic, or predicate logic in that it separates function from time and only introduces as much syntax as is necessary to deal with stabilisation behaviour. It is a purely propositional system but has second-order expressiveness. One and the same Boolean function can be represented in various ways as a PST formula, giving rise to different *timing models* which associate different stabilisation delays with different parts of the functionality and adjust the granularity of the data-dependency of delays within wide margins. We show how several standard timing analyses can be characterised as algorithms computing *correct* and *exact* stabilisation bounds for particular PST timing models. Specifically, the existence of a PST specification style for static sensitization solves the open exactness problem for this type of analysis. By choosing other timing models we can characterise timing analyses for which no algorithms so far exist. Translations between different timing models are the semantic basis for combining timing analyses.

This work puts forward an application of intuitionistic modal logic that exploits the model-theoretic strength of the constructive approach. It contrasts with the traditional point of view that focuses on the proof-theoretic aspects of intuitionistic logic.

Keywords: intuitionistic logic, modal logic, timing analysis, stabilization, combinational systems

1 Introduction

The search for new hardware timing analysis techniques is driven by two competing goals: efficiency and precision. Increasing the efficiency of an analysis algorithm means reducing the cost of designing a circuit, while increasing its precision improves the performance of the circuit itself. It is evident that there is a trade-off between both goals. Thus, it is not surprising that existing work on timing analysis, specifically of combinational circuits which will concern us here, encompasses a large variety of specialised algorithms designed for different timing models at different levels of abstraction.

The simplest and oldest known method is the *topological* analysis, which computes the length of the longest path through the circuit. It can be computed efficiently in linear time by a standard graph-theoretic algorithm. The precision of the topological delay model, however, for state-of-the-art hardware often is not acceptable since it yields a gross overestimation of the actual delay. As was pointed out in [2] optimising a circuit for speed in terms of the topological delay may actually deteriorate its performance. The obvious defect of the topological analysis is that it completely ignores functionality, *i.e.* the data-dependency of delays. For a timing analysis to be

M. Mendler: Characterising Timing Analyses in Intuitionistic Modal Logic.
Logic Journal of the IGPL, Vol 8, No. 6, pp. 821-853, Oxford University Press, 2000

adequate for state-of-the-art circuit designs a data-dependent timing model must be used [17]. An extreme case is *wave pipelining* [14], a digital design style in which the timing model must get close to the analog electrical behaviour in order to be useful.

Exact timing analysis for combinational circuits is NP-complete [23]. Practical analyses, therefore, often are based on heuristics which maintain only approximate timing information. The timing models found in the literature are quite different in the degree of data-dependency and operational modes that they consider. The main types of timing models are *transition delay* [4, 8], *delay by sequences of vectors* [17], *floating mode* [4, 7], *viability mode* [23], *static sensitization* [2], and several forms of *dynamic sensitization* [9, 30, 22, 32]. These various forms of timing analyses are designed with a view on algorithms and data structures with the consequence that the existing classification is based primarily on the method by which the delay is computed. This neglects the importance of characterising timing analyses also in semantic terms, *i.e.* determining *what* is computed rather than *how* this is done. Specifically, the following semantic questions deserve to be addressed:

1. Given the timing analysis X performed on circuit C produces the delay number δ, then what information does δ give us about the behaviour of C?
2. How do two different timing analyses X and Y relate to each other, how can we compare their relative precision?

The first question is essentially the issue of correctness and completeness of a given timing analysis. Though the existence of correctness and completeness results usually is an important lynch-pin for program analysis methods it is rarely put up in the area of hardware timing analysis. This may sometimes be the case just because the algorithm is considered to be simple enough and well understood, and sometimes because the algorithm's semantic implications are too nontrivial to be made explicit easily. An example of the latter is static sensitization analysis. It is still unclear for which classes of circuits and under which operating conditions static delay analysis, which has received quite some attention [2, 15, 35, 36], is correct and complete. The second question is the issue of semantic abstraction and refinement. It is known that different timing analyses have different relative exactness, due to varying delay models and assumptions on operating conditions. Lacking a common semantic basis different analyses are hard to compare in terms of their relative precision, which sometimes leads to paradoxical results [34]. There are purported "exact" methods which are not exact, and purported new analyses that coincide with already existing ones. Attempts to classify timing analyses exist, such as [17, 5, 33] based on *path sensitization criteria*, but these are not systematic and essentially of an algorithmic rather than a semantic nature. Yet, a semantic approach is a prerequisite to answering the second question since the relative precision of two timing analyses X and Y may depend in particular on the functionality of the circuit. So, for some class of circuits analysis X may be more exact than Y, while for another analysis Y produces tighter results.

This paper proposes to use a logic framework to answer such semantic questions. The idea is to use logic formulas to characterise the amount of semantic information about the combined temporal and functional behaviour that a given timing analysis is capable of handling in a correct and exact way. By viewing the algorithm as a formal calculus the correctness and exactness of the algorithm relative to a specific timing model can be phrased simply and rigorously as soundness and completeness of the calculus for a specific logic theory.

What logic should we be using? Clearly, it must be sufficiently expressive to capture the desired degree of functional and temporal information. If it is too weak it does not allow us to make enough distinctions. On the other hand, if the logic is too expressive, then the formalism does not contain enough structure and the classification based on it becomes uninteresting. There is a certain trade-off to fix. So when it comes to it, what is the basic semantic property that we need to express? The information we obtain from the successful execution of a timing analysis algorithm concerns *bounded stabilisation*, *i.e.* statements like "*there exists a time bound δ such that in all executions of the system the distance between stabilisation of signals a and b is at most δ.*" Since bounded stabilisation is a property of sets of infinite waveforms (or traces) and requires quantification over waveforms we need second-order expressiveness in our logic. This rules out well-known classical logics such as propositional temporal and modal logics, or first-order predicate logic. On the other hand, second-order predicate logic in which this can be expressed seems to be too general and therefore logic overkill for the simple purpose of expressing stabilisation behaviour of purely combinational systems. Fortunately, we can do better by taking a more dedicated route. We exploit the observation that by using an intuitionistic rather than classical approach bounded stabilisation can be expressed by purely propositional means. We introduce an intuitionistic modal theory called *Propositional Stabilisation Theory*, or PST for short, which combines the semantic expressiveness of second-order predicate logic with the syntactic economy of propositional logic. We show that several standard timing analyses for combinational circuits can be classified naturally in terms of correctness and completeness for characteristic PST specification styles. In particular, a PST timing model for static sensitization is presented. This solves the open exactness problem for static sensitization analysis, and provides a rigorous uniform framework in which different timing analyses may be combined. Moreover, we show that PST has considerable expressiveness, which suggests that the framework captures many interesting, but yet unknown, timing analyses with different granularity of timing information.

Before we start with the technical details let us stress that although this paper is biased towards digital circuits the application of PST reported herein is not limited to hardware. It covers equally well the analysis of stabilisation behaviour for software, more specifically of finite combinational input-output systems, where *combinational* refers to the property that all internal states are transient. Such systems arise frequently, notably in data-flow programming [39].

2 The Intuitionistic Modal Theory PST

We obtain PST as a particular semantic interpretation of intuitionistic propositional logic extended by a modal operator $\bigcirc$ with the axioms

$$\begin{array}{lcl} \bigcirc I & : & \varphi \supset \bigcirc\varphi \\ \bigcirc M & : & \bigcirc\bigcirc\varphi \supset \bigcirc\varphi \\ \bigcirc S & : & (\bigcirc\varphi \wedge \bigcirc\psi) \supset \bigcirc(\varphi \wedge \psi) \end{array}$$

and the rule $\varphi \supset \psi \Rightarrow \bigcirc\varphi \supset \bigcirc\psi$. This system is known as Propositional Lax Logic (PLL) [12] or Computational Logic [3]. The modal operator $\bigcirc$ which has been introduced originally by Curry [6] arises under many different names in Mathematics and Computer Science. In the latter community its most well-known appearance is as a *strong monad* in the work of Moggi [28] where $\bigcirc$ is used as a type-theoretic operator for notions of computation. This paper, like [12], takes a logical and model-theoretic perspective on the intuitionistic modality. From a modal logic point of view $\bigcirc$ is rather unusual. For instance, $\bigcirc I$ is part of a S4-type possibility while $\bigcirc S$ is typical for a standard necessity. On the other hand, $\bigcirc S$ is never adopted for possibility and $\bigcirc I$ never for necessity. Then again, both the axiom $\bigcirc M$ and the rule $\varphi \supset \psi \Rightarrow \bigcirc\varphi \supset \bigcirc\psi$ express properties of both S4-type possibility and necessity. It turns out that $\bigcirc$ does not have a *classical* Kripke semantics, neither for the possibility nor the necessity interpretation. However, perhaps surprisingly, it does have a natural *intuitionistic* semantics. An adequate intuitionistic Kripke style model theory for PLL is developed in [12], based on so-called Kripke *constraint models*. Other types of Kripke style models for PLL are the *$\mathcal{J}$-frames* and *$\mathcal{J}$-spaces* of [13]. A more general algebraic semantics for PLL can be provided by Heyting algebras with a *modal operator* [19].

What is the semantic intuition behind the modal operator $\bigcirc$? According to the interpretation that we wish to put forward and support with this work the modal operator $\bigcirc$ formalises a relaxed notion of correctness according to which $\bigcirc\varphi$ means "*φ holds up to a constraint.*" Such relativised statements occur frequently in the formal specification and verification of behavioural abstractions, both for software and for hardware. Under this reading the three axioms $\bigcirc I, \bigcirc M, \bigcirc S$ reflect the three characteristic operations on constraints, specifically $\bigcirc I$ the *trivial constraint*, $\bigcirc M$ *sequential composition*, and $\bigcirc S$ the *parallel composition* of constraints. The special theory PST that we will be interested in here, arises as a more specific semantic interpretation for which a modalised formula $\bigcirc\varphi$ comes down to the statement "*φ holds up to bounded stabilisation.*" As we will see, in PST the three axioms correspond to the three operations of the max-plus algebra $(\mathbb{N}, 0, +, \mathit{max})$ [1], which is the algebraic basis of (upper bound) timing analysis. The axiom $\bigcirc I$ corresponds to the zero delay 0, $\bigcirc M$ to addition $+$, and $\bigcirc S$ to the maximum operation *max* on natural numbers. In the timing semantics of PST the modality $\bigcirc$ is somewhat more natural in that it specialises to a form of intuitionistic possibility.

The theory PST will be presented as a realisability style interpretation of PLL as opposed to a class of modal Heyting algebras or a class of Kripke constraint models. This provides for an intensional semantics in which quantitative timing information can be represented directly, *viz.* by realisers. It can be shown, however, that if we abstract from the realisers the resulting extensional semantics can be captured equivalently in terms of a class of Kripke constraint models. An indication of this will be given in Section 4.2. A more detailed presentation of this extensional semantics can be found in [25], albeit for a slightly more restrictive setting.

2.1 Syntax

The *formulas* of PST are generated by the language

$$\varphi \quad ::= \quad \mathit{true} \mid \mathit{false} \mid a = 0 \mid a = 1 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \supset \varphi \mid \bigcirc\varphi,$$

where a ranges over a countably infinite set $\mathbb{S} = \{a, b, c, \dots\}$ of *signal names.* Formulas $a = 0$ and $a = 1$ are propositional atoms representing the primitive statements "*signal a is stable* 1" and "*signal a is stable* 0," respectively. From these primitive assertions complex statements of stabilisation behaviour may be built up using the logic connectives of PST. We will later introduce further atomic propositions like $a = \frac{1}{2}$ or $a = E$, where E is a Boolean expression over signals, but these will not increase the expressiveness. It would also be possible to extend the formalism to arbitrary finite value domains $\mathbb{D}$ using atomic sentences $a = v$, $v \in \mathbb{D}$ with the obvious interpretation. For simplicity, however, we restrict ourselves to Boolean-valued signals. Also note that although the constants *true*, *false* are redundant, it will be convenient to consider them as primitives. The constant *false* can be represented as $a = 1 \land a = 0$, and *true* by $a = 1 \supset a = 1$, both for arbitrary $a \in \mathbb{S}$. We introduce negation $\neg\varphi$ as an abbreviation for $\varphi \supset \mathit{false}$, and $\varphi \equiv \psi$ denotes bi-implication $(\varphi \supset \psi) \land (\psi \supset \varphi)$.

2.2 Semantics

The basic elements of our semantics are *signals*, *waveforms*, *stabilisation bounds*, and *behaviours*. A *signal* is a function from time to values, $s \in \mathbb{N} \to \mathbb{B}$, time being represented by the natural numbers and values by Booleans $\mathbb{B} = \{0, 1\}$. In a more general setup a signal might be a function $\mathbb{N} \to \mathbb{D}$ where $\mathbb{D}$ is some (finite) value domain. Signals will be the semantic denotation of signal names. A *waveform* is a function that maps every signal name to a signal, *i.e.* a function $V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B}$. These will play the rôle of semantic valuations of formulas.

When it comes to timing analysis we are concerned with not merely *whether* a set of waveforms satisfies a PST formula φ but also *how* it achieves this. The intensional degree of validity is the timing and it is measured in terms of stabilisation bounds. It depends on the formula how much quantitative timing information is implied with it. To make this explicit we associate with every formula φ a set $|\varphi|$ of *stabilisation bounds* as follows:

$$\begin{array}{rcl}
|\mathit{false}| & = & \underline{1} \;=\; |\mathit{true}| \\
|a = 1| & = & \underline{1} \;=\; |a = 0| \\
|\varphi \land \psi| & = & |\varphi| \times |\psi| \\
|\varphi \lor \psi| & = & |\varphi| + |\psi| \\
|\varphi \supset \psi| & = & |\varphi| \to |\psi| \\
|\bigcirc\varphi| & = & \mathbb{N} \times |\varphi|,
\end{array}$$

where $\underline{1} = \{0\}$ is a distinguished singleton set. More generally, we will use the notation $\underline{n}$ for $n \in \mathbb{N}$ to denote the set $\{0, 1, \dots, n-1\}$, discretely ordered. We identify $\mathbb{B}$ and $\underline{2}$. As usual the elements of the disjoint sum $|\varphi| + |\psi|$ are pairs $(0, c)$ where $c \in \varphi$ or $(1, d)$ where $d \in \psi$. An element $c \in |\varphi|$ is called a *stabilisation bound* or simply a *bound* for φ. Note that $|\varphi|$ always is non-empty, so that every formula has at least one bound. We say that a waveform $V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ *validates* a formula φ with bound $c \in |\varphi|$, written $V \models c : \varphi$, according to the semantic clauses

$$\begin{array}{llllll}
V & \models & 0 : \mathit{true} & & \\
V & \models & 0 : a = 1 & \mathit{iff} & V(a) \downarrow_0 1 \\
V & \models & 0 : a = 0 & \mathit{iff} & V(a) \downarrow_0 0 \\
V & \models & (c, d) : \varphi \wedge \psi & \mathit{iff} & V \models c : \varphi \text{ and } V \models d : \psi \\
V & \models & (0, c) : \varphi \vee \psi & \mathit{iff} & V \models c : \varphi \\
V & \models & (1, d) : \varphi \vee \psi & \mathit{iff} & V \models d : \psi \\
V & \models & f : \varphi \supset \psi & \mathit{iff} & \text{for all } \delta \in \mathbb{N} \text{ and } c \in |\varphi|, \\
 & & & & \quad \text{if } V^\delta \models c : \varphi \text{ then } V^\delta \models f\, c : \psi, \\
V & \models & (\delta, c) : \bigcirc\varphi & \mathit{iff} & V^\delta \models c : \varphi,
\end{array}$$

where $V(a) \downarrow_t v$ means that signal a in waveform V stabilises to value v at time t, *i.e.* $\forall s \geq t.\, V(a)(s) = v$, and V^δ is the *time-shift* $V^\delta(a)(t) = V(a)(t + \delta)$ of V. This operation is lifted to sets of waveforms in the standard way.

Our semantics associates with every pair $c : \varphi$ consisting of a formula φ and a bound $c \in |\varphi|$ a waveform set $[\![c : \varphi]\!] = \{ V \mid V \models c : \varphi \}$. It is useful to view $c : \varphi$ as a new kind of formula which represents the refinement of φ by intensional stabilization information c. The colon then becomes a binary connective that separates the intensional from the extensional aspect. We will give more details later about what kind of waveform sets can be specified in this way. At this point it suffices to mention that all sets $[\![c : \varphi]\!]$ have the following *time invariance* property: If $V \in [\![c : \varphi]\!]$ then $V^\delta \in [\![c : \varphi]\!]$, too, for all $\delta \in \mathbb{N}$. Time invariance is a feature of stabilization properties. If $V \in [\![c : \varphi]\!]$ but $V^\delta \notin [\![c : \varphi]\!]$ for some $\delta > 0$, then the information expressed by $c : \varphi$ would only be a transient feature of V and thus does not count as a stabilization property. In the following we will refer to time invariant subsets of waveforms as (stabilisation) *behaviours*.

In a concrete timing analysis problem we are given some behaviour C (of an implementation) and a formula φ (as its specification) and ask for a stabilisation bound c such that $C \subseteq [\![c : \varphi]\!]$. If such a bound exists we say that C is *well timed* for φ with bound c, and write $C \models c : \varphi$. In general there will be infinitely many c for which this is the case. We will be interested in optimal bounds. To make this formal we introduce a partial ordering $\sqsubseteq$ on bounds, so that $c \sqsubseteq d$ means c is *tighter* than d. In this way, the partial ordering $|\varphi|$ measures the intensional stabilization information that is associated with φ. The ordering on $|\varphi|$ is generated by induction on φ from the natural ordering $\leq$ on $\mathbb{N}$, taking point-wise ordering on products $|\varphi| \times |\psi|$ and function spaces $|\varphi| \to |\psi|$. For disjoint unions $|\varphi| + |\psi|$ we take the discrete ordering, so that $(i, c) \sqsubseteq (j, d)$ *iff* $i = j$ and $c \sqsubseteq d$. Then, a stabilisation bound $c \in |\varphi|$ is *exact* or *worst-case* for C and φ, if for all $d \in |\varphi|$ such that $d \sqsubseteq c$ we have $c = d$ *iff* $C \models d : \varphi$. The following monotonicity property highlights the intuitionistic nature of our semantics:

Proposition 2.1
Let C, D be behaviours and $c, d \in |\varphi|$ such that $D \subseteq C$ and $c \sqsubseteq d$. Then, $C \models c : \varphi$ implies $D \models d : \varphi$.

There are two useful classes of formulas for which the relation $\sqsubseteq$ has special properties relating it to the realisability semantics. The first is a particularly simple one, the class of formulas for which $|\varphi|$ is (order) isomorphic to $\underline{1}$. Such φ are called *non-informative* since they only carry trivial stabilization information. The symbol ζ will be used to range over non-informative formulas. The other class of formulas are the *elementary* formulas, ranged over by θ. Elementary formulas are generated by the grammar

$$\theta \quad ::= \quad \mathit{true} \mid \mathit{false} \mid a = 1 \mid a = 0 \mid \zeta \mid \theta \wedge \theta \mid \bigcirc\zeta \mid \varphi \supset \theta,$$

where ζ is non-informative and φ arbitrary. Note that every non-informative formula is elementary. The following Proposition 2.2 provides the basis for the timing analysis discussed in Section 3. It implies the existence of unique worst-case stabilization bounds for elementary formulas.

Proposition 2.2
Let θ be an elementary formula and C a behaviour such that C is well-timed for θ. Then, the set $\{ c \mid C \models c : \theta \}$ ordered by $\sqsubseteq$ is (nonempty and) a complete lower semilattice.

Note that Proposition 2.2 does not hold for arbitrary formulas. Consider $\varphi = \bigcirc\bigcirc(a = 1)$ and a waveform V that switches a to 1 exactly at time point 10, and leaves it 0 until then, *i.e.* $V(a)(t) = 1$ *iff* $t \geq 10$. Now, working out the semantics we find that $V \models (r, (s, 0)) : \varphi$ *iff* $r + s \geq 10$. For instance, $V \models (0, (10, 0)) : \varphi$ and $V \models (10, (0, 0)) : \varphi$. However, the only stabilization bound that is both below $(10, (0, 0))$ and $(0, (10, 0))$ in the $\sqsubseteq$ ordering is $(0, (0, 0))$, but $V \not\models (0, (0, 0)) : \varphi$. Hence, the set $\{ c \mid C \models c : \varphi \}$ (ordered by $\sqsubseteq$) where $C = \{ V^{\delta} \mid \delta \in \mathbb{N} \}$ is not a lower semilattice. Note that, *e.g*, the orderings $(|\varphi| + |\psi|, \sqsubseteq)$ of stabilization bounds for disjunctions do not form semilattices either.

If we abstract from specific stabilization bounds and only concern ourselves with whether or not a behaviour is well timed for a formula we obtain a notion of extensional validity that links behaviours and formulas. We write $C \models \varphi$ if there exists $c \in |\varphi|$ with $C \models c : \varphi$. PST, then, is the set of all formulas φ such that $C \models \varphi$ for all behaviours C, *i.e.* the formulas well timed for all behaviours. Because of monotonicity 2.1 this is the same as saying that φ is well timed for the set $\mathbb{S} \to \mathbb{N} \to \mathbb{B}$ of all waveforms. Thus,

$$\varphi \in \mathrm{PST} \quad \mathit{iff} \quad \exists c \in |\varphi|.\, \forall V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B}.\, V \models c : \varphi.$$

This is essentially a set-theoretic Medvedev style realisability interpretation for PLL with stabilisation bounds as realisers. Besides the extra modality operator there are two main variations here to Medvedev's realisability interpretation [24] of intuitionistic logic. On the one hand our semantics is more specific in that it uses a fixed choice of singleton sets $|a = 1| = |a = 0| = \underline{1}$ for the propositional atoms. Medvedev's interpretation quantifies over all interpretations that associate arbitrary finite sets of “problems” with propositional atoms. In another direction our semantics is more general. Medvedev applies a classical reading of implication whereby $\varphi \supset \psi$ is realised by a function $f \in |\varphi| \to |\psi|$ for valuation V if $\forall c \in |\varphi|.\, V \models c : \varphi \Rightarrow V \models fc : \psi$. In our semantics, we require this to hold not only of V but also of all its time shifts V^{δ}. This amounts to an intuitionistic reading of realisability on waveforms V as linear Kripke models. For constant waveforms, *i.e.* in which no signal changes, we get back Medvedev's classical realisability. To sum up, our semantic definition of PST (ignoring the modality) may be thought of as an *intuitionistic version* of a Medvedev style semantics of *singleton problems* on *linear* Kripke models. This relationship will be invesigated more closely in Section 4.3. For a systematic study of Medvedev's logic of singleton problems the reader is referred to the paper [27]. A survey on notions of realisability can be found in [37]. For our purposes the realisability approach suggests itself as being expedient to separate the two concerns of *function*, represented by formulas φ, and *timing*, represented by stabilisation bounds c as realisers.

PST is a purely propositional theory and even though it does not have quantification and time variables it can deal with timing and bounded stabilisation. The complexity, of course, resides in its intuitionistic semantics. This semantics obtains a translation of PST into a fragment of classical higher-order predicate logic. Let us make this more precise and then take a look at some examples. Consider the following "standardization" translation of a pair $x : \varphi$ into a typed predicate logic formula $[x : \varphi]_t$ relative to time t:

$$\begin{array}{rcl}
[x : a = 1]_t & = & \forall s \geq t.\, a(s) = 1 \\
[x : a = 0]_t & = & \forall s \geq t.\, a(s) = 0 \\
[x : \mathit{false}]_t & = & \mathit{false} \\
[x : \mathit{true}]_t & = & \mathit{true} \\
[x : \varphi \wedge \psi]_t & = & [\pi_1 x : \varphi]_t \wedge [\pi_2 x : \psi]_t \\
[x : \varphi \vee \psi]_t & = & \forall y \in |\varphi|.\, x = (0, y) \Rightarrow [y : \varphi]_t \wedge \forall y \in |\psi|.\, x = (1, y) \Rightarrow [y : \psi]_t \\
[x : \varphi \supset \psi]_t & = & \forall s \geq t.\, \forall y \in |\varphi|.\, [y : \varphi]_s \Rightarrow [x\, y : \psi]_s \\
[x : \bigcirc\varphi]_t & = & [\pi_2 x : \varphi]_{t+\pi_1 x}.
\end{array}$$

Observing that $[x : \varphi]_t$ is only little more than the predicate logic formalisation of the semantic conditions of $\models$ it is not difficult to show the following proposition:

Proposition 2.3
$V \models c : \varphi$ *iff* $[c : \varphi]_0$ is classically valid, where all signal names a are interpreted by their associated signal functions $V(a)$.

The following examples illustrate how in PST a specification $c : \varphi$ conveniently separates the timing and the functional aspects, which in a conventional timing diagram or an equivalent predicate logic formalisation are intertwined.

Example 2.4
Consider the formula $\varphi \equiv_{df} (a = 0 \vee a = 1) \supset \bigcirc(b = 0)$. What does it mean for a behaviour C to be well-timed for φ? We find that $|\varphi| = (\underline{1} + \underline{1}) \to \mathbb{N} \times \underline{1}$, which is (order) isomorphic to $\mathbb{B} \to \mathbb{N}$. Thus, a stabilisation bound for φ, up to isomorphism, is a pair of natural numbers. It can be shown that $C \models \varphi$ *iff* there exists a stabilisation bound $f \in \mathbb{B} \to \mathbb{N}$ so that whenever a stabilises to value v at some time t then b will stabilise to 0 with maximal delay $f(v)$. Formally, $\exists f \in \mathbb{B} \to \mathbb{N}.\, \forall V \in C.\, \forall t \in \mathbb{N}.\, \forall v \in \mathbb{B}.\, V(a) \downarrow_t v \Rightarrow V(b) \downarrow_{t+f(v)} 0$. This is a second-order timing condition on C. Note that the delay $f(v)$ is data-dependent. We call formulas of the form $(\bigvee_i \bigwedge_j s_{ij} = v_{ij}) \supset \bigcirc(s = v)$ *transitions.*

Example 2.5
All theorems of PLL are theorems of PST. Take the PLL axiom $\bigcirc S \equiv_{df} (\bigcirc(a = 1) \wedge \bigcirc(b = 0)) \supset \bigcirc(a = 1 \wedge b = 0)$, for instance. Its set of stabilisation bounds is

$|\bigcirc S| = ((\mathbb{N} \times \underline{1}) \times (\mathbb{N} \times \underline{1})) \to (\mathbb{N} \times (\underline{1} \times \underline{1}))$ which is isomorphic to $(\mathbb{N} \times \mathbb{N}) \to \mathbb{N}$. To say $\bigcirc S$ is a theorem means that there exists a function $f \in (\mathbb{N} \times \mathbb{N}) \to \mathbb{N}$ such that for all waveforms $V \models f : \bigcirc S$. It is not difficult to show that the tightest, *i.e.* least in the $\sqsubseteq$ ordering, such f is the maximum operation *max* on $\mathbb{N}$. Similarly one shows that the tightest stabilisation bound for $\bigcirc I \equiv_{df} a = 1 \supset \bigcirc(a = 1)$, up to isomorphism, is the constant zero 0, and for $\bigcirc M \equiv_{df} \bigcirc\bigcirc(a = 1) \supset \bigcirc(a = 1)$ addition $+$ on $\mathbb{N}$. In this way the three arithmetic operations of the max-plus algebra $(\mathbb{N}, 0, max, +)$ are characterised as stabilisation bounds of PST. Again, validity of $\bigcirc S, \bigcirc I, \bigcirc M$ is a higher-order condition that cannot be expressed by standard (classical) propositional temporal or modal logics.

Before we discuss timing analyses in the next Section 3 it will be useful to introduce some derived constructions of PST, which give us the stationary state, ternary signal algebra, and dynamic choice. Further meta-theoretic results about PST of more general interest will be given in Section 4 later.

2.3 Double Negation and the Stationary State

Double negation in PST specifies the stationary state. First note that doubly negated formulas $\neg\neg\varphi$ are non-informative. The associated set of stabilization bounds $|\neg\neg\varphi| = (|\varphi| \to \underline{1}) \to \underline{1}$ is isomorphic to $\underline{1}$, so that $|\neg\neg\varphi|$ consists of a single canonical element 0. It turns out that we can obtain the semantics of $0 : \neg\neg\varphi$ by interpreting φ as a classical statement about the stationary state in which an atomic proposition $a = v$ is read as "signal a stabilises to v eventually" and the modal operator $\bigcirc$ is dropped.

To be more precise, let $\mathbb{K} = \{0, \frac{1}{2}, 1\}$ be the three-element Kleene set. The *stationary state* assumed by a waveform $V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ is the three-valued valuation $V^\infty \in \mathbb{S} \to \mathbb{K}$ given by $V^\infty(a) = v \in \mathbb{B}$ if $\exists t.\, V(a) \downarrow_t v$, and $V^\infty(a) = \frac{1}{2}$ otherwise. Let us write $V^\infty \models_c \varphi$ if φ is a classically true propositional statement, where all atomic $a = 0$, $a = 1$ are replaced by $V^\infty(a) = 0$, $V^\infty(a) = 1$, respectively, and all subformulas $\bigcirc\psi$ by ψ.

PROPOSITION 2.6
$V \models 0 : \neg\neg\varphi$ *iff* $V^\infty \models_c \varphi$.

We will refer to the set of ternary valuations $\{\, V^\infty \in \mathbb{S} \to \mathbb{K} \mid V^\infty \models_c \varphi \,\}$ as the *stationary behaviour* specified by $\neg\neg\varphi$.

EXAMPLE 2.7
The formula $\neg\neg\varphi = \neg\neg((a = 0 \wedge b = 1) \vee (a = 1 \wedge b = 0))$ specifies the stationary state of an ideal inverter: both a and b stabilise eventually to opposite values. Formally, the stationary behaviour specified by $\neg\neg\varphi$ is specified by the condition $V^\infty(a) \in \mathbb{B}$ & $V^\infty(a) = \overline{V^\infty(b)}$. The formula $\mathsf{osc}(a) \equiv_{df} \neg(a = 1 \vee a = 0)$ says that a oscillates, *i.e.* the stationary value of a is $\frac{1}{2}$, and $\mathsf{stab}(a) \equiv_{df} \neg\neg(a = 0 \vee a = 1)$ means that eventually a stabilises to 0 or to 1.

2.4 Encoding Ternary Signal Algebra

Let the language of PST be extended by new atomic propositions of the form $a = E[b_1, \ldots, b_n]$ where $a \in \mathbb{S}$ and E a Boolean expression[1] in the signals $b_1, \ldots, b_n \in \mathbb{S}$. We use the notation $E[b_1, \ldots, b_n]$ to indicate that $b_1, \ldots, b_n$ are *all* the signals that occur in E. The intended meaning of $a = E$ is "*a is constant v, where v is the value of expression E in the stationary state.*" We add such new atoms as abbreviations in the following way:

$$a = E[b_1, \ldots, b_n] \quad \equiv_{df} \quad \bigwedge_{\vec{v} \in \mathbb{K}^n} (\neg\neg(b_1 = v_1 \wedge \cdots \wedge b_n = v_n)) \supset a = E[v_1, \ldots, v_n],$$

where $E[v_1, \ldots, v_n] \in \mathbb{K}$ denotes the ternary evaluation of the Boolean expression E, and $s = \frac{1}{2}$ abbreviates *true*. The new atoms $a = E$ are non-informative, *i.e.* $|a = E|$ is isomorphic to $\underline{1}$. They indeed have the right semantics:

Proposition 2.8
$V \models 0 : a = E[b_1, \ldots, b_n]$ *iff* $V \models 0 : a = E[V^\infty(b_1), \ldots, V^\infty(b_n)]$.

Observe that the definition of $a = E$ includes the original primitives $a = 1$ and $a = 0$ as special cases, if we consider the constants $1, 0$ as Boolean expressions over an empty list of signals.

Example 2.9
We find that $c = \bar{a}$, which is equivalent to $(\neg\neg a = 1 \supset c = 0) \wedge (\neg\neg a = 0 \supset c = 1)$, states that if a becomes stationary with value $v \in \mathbb{B}$ then signal c is constant at $\bar{v}$. The special case $a = a \equiv_{df} (\neg\neg a = 1 \supset a = 1) \wedge (\neg\neg a = 0 \supset a = 0)$ means that a is stationary, *i.e.* it is either constant or it oscillates forever. If we exclude oscillation with $\mathsf{stab}(a) \equiv_{df} \neg\neg(a = 0 \vee a = 1)$, the formula $\mathsf{const}(a) \equiv_{df} a = a \wedge \mathsf{stab}(a)$ expresses that a is *constant*.

It is crucial to interpret E in $a = E$ as a three-valued expression as opposed to a two-valued one, for otherwise the semantics would be unsound. For example, although in Boolean algebra $b \cdot \bar{b}$ is identical to 0, the atomic propositions $a = b \cdot \bar{b}$ and $a = 0$ are different: The former means "*a is constant* 0 *if b stabilises*" whereas the latter says "*a is constant* 0" which is stronger. The difference is important as there is no guarantee that b ever stabilises, in general. Formally, this is taken care of in three-valued Kleene algebra, where $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}$, which is different from 0.

With the derived "equation-like" atomic propositions $s = E$ we can embed the three-valued Kleene algebra into PST. These expressions behave like ternary expressions in the Kleene algebra $\mathbb{K}$. In particular they enjoy the extensionality property, *i.e.* if $E_1[b_1, \ldots, b_n]$ and $E_2[b_1, \ldots, b_n]$ denote the same three-valued function (of the b_i), then $a = E_1$ is semantically equivalent to $a = E_2$. Moreover they are substitutive, *i.e.* the formula scheme $(a = E \wedge b = F) \supset (b = F\{E/a\} \wedge a = E\{F/b\})$ is a PST theorem. Boolean algebra is obtained as a special case by adding the axioms $\mathsf{stab}(s)$, for all $s \in \mathbb{S}$.

[1] Actually an expression in the ternary algebra $\mathbb{K}$, which is the same as a Boolean expression with ternary semantics.

2.5 Static Versus Dynamic Choice

An important feature of our semantics (and of every other realizability semantics of intuitionistic logic) is the ordering in the quantification over stabilization bounds (realisers) and waveforms (valuations). We have $C \models \varphi$ *iff* $\exists c \in |\varphi|.\, \forall V \in C.\, V \models c : \varphi$, *i.e.* the stabilization bound c must be uniform for all waveforms $V \in C$. From the timing viewpoint we might say that the bound c is chosen *statically* for C. This contrasts with a *dynamic* choice that would result from interchanging the two quantifiers, *i.e.* from taking $\forall V \in C.\, \exists c \in |\varphi|.\, V \models c : \varphi$ as the notion of validity. Now the choice for c may depend on the individual waveform $V \in C$. Let this notion of validity be denoted by $C \models_d \varphi$. It is evident that $\models_d$ is a less restrictive semantics, *i.e.* $C \models \varphi$ implies $C \models_d \varphi$. The simple examples below will show that it is properly weaker. It turns out that this weaker dynamic reading can be obtained by systematically replacing all modal operators by double negation and all disjunctions $\vee$ by the derived binary operator $\oplus$, defined as

$$\varphi \oplus \psi \quad \equiv_{df} \quad ((\varphi \supset \psi) \supset \psi) \wedge ((\psi \supset \varphi) \supset \varphi).$$

Proposition 2.10
We have $C \models_d \varphi$ *iff* $C \models \varphi^d$ where φ^d is obtained by replacing all occurrences of sub-formulas $\bigcirc\psi$ in φ by $\neg\neg\psi$ and all occurrences of $\psi_1 \vee \psi_2$ by $\psi_1 \oplus \psi_2$.

In the sense made clear by Proposition 2.10 we may view $\oplus$ as the dynamic version of disjunction $\vee$ and $\neg\neg$ as the dynamic version of $\bigcirc$. Note that the transformed φ^d is non-informative, *i.e.* its set of stabilization bounds is isomorphic to $\underline{1}$. This means that the semantics basically collapses all stabilization bounds in this case.

Example 2.11
If a behaviour C is to be well timed for $a = 0 \vee a = 1$ there must be $c \in |a = 0 \vee a = 1| = \underline{1} + \underline{1}$ such that $C \models c : a = 0 \vee a = 1$. Depending on whether $c = (0, 0)$ or $c = (1, 0)$ this implies $C \models 0 : a = 0$ or $C \models 0 : a = 1$. Thus, $C \models a = 0 \vee a = 1$ *iff* all waveforms $V \in C$ have signal a constant 0 or all $V \in C$ have a constant 1. To achieve this C is forced to make a static decision between $a = 0$ and $a = 1$ and stick to it for all its constituent waveforms. In contrast with this consider the proposition $a = 0 \oplus a = 1$ which by definition is $((a = 0 \supset a = 1) \supset a = 1) \wedge ((a = 1 \supset a = 0) \supset a = 0)$. We may use Proposition 2.10 to conclude that $C \models a = 0 \oplus a = 1$ *iff* for all $V \in C$ there exists $c \in |a = 0 \vee a = 1| = \underline{1} + \underline{1}$ such that $V \models c : a = 0 \vee a = 1$. This means that for every $V \in C$, $V \models 0 : a = 0$ or $V \models 0 : a = 1$. In other words, $C \models a = 0 \oplus a = 1$ *iff* in every waveform $V \in C$ signal a is constant 0 or 1, so the choice is dynamic. Note that $a = 0 \oplus a = 1$ is semantically equivalent to $\mathsf{const}(a) = \neg\neg(a = 0 \vee a = 1) \wedge (\neg\neg a = 0 \supset a = 0) \wedge (\neg\neg a = 1 \supset a = 1)$ as defined in Example 2.9.

Example 2.12
A similar situation occurs with the difference between $C \models \bigcirc(a = 0)$ and $C \models \neg\neg a = 0$. The former means there exists a uniform stabilization bound for when signal a in all $V \in C$ stabilises to 0, while the latter only says that in every $V \in C$ signal a eventually stabilises to 0 (*cf.* Example 2.7). Again, this is the difference between a static or a dynamic choice, or the difference between $\exists\delta.\, \forall V \in C.\, V^\delta \models 0 : a = 0$ and $\forall V.\, \exists\delta.\, V^\delta \models 0 : a = 0$.

3 Application to Timing Analysis

We now rush on to discuss the main topic of this paper which is the application of PST to the problem of characterising the correctness and exactness of combinational timing analyses. Readers more interested in pure logic and the meta-theoretic properties of PST may wish to move on to Section 4 and perhaps come back to this section later.

Our application rests on the observation that one and the same Boolean function f may be specified in many different ways as a PST formula φ_f. With each such choice the static functional behaviour f is enriched in a characteristic way by stabilisation information. We can view φ_f as a *timing model* of f, and the *timing analysis* of a given circuit behaviour C, in a nutshell, as the problem of verifying that C is well timed for φ_f, and finding a tightest-fitting stabilisation bound $c \in |\varphi_f|$ such that $C \subseteq [\![c : \varphi_f]\!]$. The point is that with different choices of $f \mapsto \varphi_f$ it is possible to adjust the granularity and amount of extra timing information, and in this fashion characterise different types of timing analyses.

It is important to observe that our characterisation game is nontrivial because of the semantic gap between φ and $\neg\neg\varphi$. While two timing models φ_f and ψ_f may express the same stationary behaviour f, *i.e.* $\neg\neg\varphi_f \equiv \neg\neg\psi_f$, the included transient properties may differ, *i.e.* $\varphi_f \not\equiv \psi_f$. Here we exploit the intuitionistic nature of the stabilisation semantics of PST. In a classical setting both φ and $\neg\neg\varphi$ would coincide and our programme collapse. The rich semantic range of behaviours between φ and $\neg\neg\varphi$ will be illustrated in Section 3.1 below.

To link up with the standard way of representing behaviours we need a few additional notions. A *function unit* (*fu*) is given by a triple $F = (I, O, f)$, with *inputs* $I = \{a_1, \ldots, a_l\} \subseteq \mathbb{S}$, *outputs* $O = \{b_1, \ldots, b_m\} \subseteq \mathbb{S}$, and Boolean function $f \in \mathbb{B}^l \to \mathbb{B}^m$. F may represent a simple gate like AND, OR, INV, a complex gate such as a multiplexor, or a whole combinational circuit. More generally, a function unit may represent any finite combinational input-output system, either by bit-vector coding or by replacing $\mathbb{B}$ by some finite value domain $\mathbb{D}$. Also, we could let f be a partial Boolean function, or a ternary relation on the signals $I \cup O$, so as to capture circuits with internal feedback and potential oscillatory behaviour. The term “function unit” is chosen to stress the connections with data-flow programming.

Definition 3.1 (Timing Model)
Let $F = (I, O, f)$ be a *fu*. We call a formula φ_F a (elementary) *timing model* of F if the stationary behaviour specified by $\neg\neg\varphi_F$ (*cf.* Sec. 2.3) on the signals $I \cup O$ coincides with the (graph of the) ternary extension of f, and if φ_F is elementary.

If φ_F is a timing model of F then the stationary behaviour captured by $\neg\neg\varphi_F$ corresponds to the functionality of F, *i.e.* the function f can be recovered completely from the stationary semantics of φ_F. The restriction that φ_F is elementary ensures that worst-case stabilization bounds for φ_F (measured by the partial ordering $\sqsubseteq$) exist.

We can now elaborate a bit further on the view that timing analysis is about establishing well timedness w.r.t. a given timing model. In practice, the waveforms that make up the behaviour C to be well timed are not given directly, but are specified themselves within PST. Typically, C is the behaviour of a composite system, *i.e.* a list $\vec{F} = F_1, \ldots, F_n$ of *fu*s, each of which is specified by a timing model φ_{F_i} and a stabilisation bound $c_i \in |\varphi_{F_i}|$, $i = 1, \ldots, n$. The behaviour C, then, is

$$C \;=\; \bigcap_{1 \le i \le n} [\![c_i : \varphi_{F_i}]\!] \;=\; [\![(c_1, \ldots, c_n) : \varphi_{F_1} \wedge \cdots \wedge \varphi_{F_n}]\!].$$

By changing the mapping $\varphi : F_i \mapsto \varphi_{F_i}$ it is possible to associate different choices of timing models with the components of the system and thus, depending on the purpose, adjust for a suitable precision in the timing description. Similarly, the specification ψ for which we want C to be well timed is not arbitrary but generated from the *fu*s as well, *i.e.* $\psi = \psi_{\vec{F}}$. Again, we may build the mapping $\psi : \vec{F} \mapsto \psi_{\vec{F}}$ in accordance with the required precision of the analysis. Typically, $\psi_{\vec{F}}$ would be a timing model for the composite Boolean function realised by the system's components $\vec{F}$. Note that the mapping ψ must translate lists $\vec{F}$ of arbitrary length to accommodate systems with arbitrary number of components. If we consider the *fu*s F_i as the input to the timing analysis, then a particular timing analysis is characterised by the choice of the timing models φ_{F_i} for the components and $\psi_{\vec{F}}$ for the composite system. Thus, a timing analysis is characterised by two mappings φ and ψ from *fu*s to timing models.

Definition 3.2 (Timing Analysis)
Let $\varphi : X \mapsto \varphi_X$ be a mapping which translates a *fu* X to a timing model for X and $\psi : \vec{Y} \mapsto \psi_{\vec{Y}}$ a mapping that determines for every list of component *fu*s a timing model $\psi_{\vec{Y}}$ for the composite system. Then, a $[\varphi, \psi]$-*style timing analysis* is a partial function $T = T[\varphi, \psi]$, that computes a stabilisation bound $T[\varphi, \psi](\vec{F}, \vec{c}) \in |\psi_{\vec{F}}|$ for every list of *fu*s $\vec{F}$ and stabilisation bounds $\vec{c} = c_1, \ldots, c_n$ such that $c_i \in |\varphi_{F_i}|$.

A timing analysis $T[\varphi, \psi]$ as described above is nothing more than a function that turns stabilisation bounds into stabilisation bounds. This cannot be all, since the value $T[\varphi, \psi](\vec{F}, \vec{c}) \in |\psi_{\vec{F}}|$ returned is not very relevant as long as it does not imply any semantic information about the real-time behaviour of the system that is analysed. To rule out trivial solutions to the timing analysis problem we must impose semantic soundness and completeness conditions. To do this in a convenient way let us abbreviate the conjunction $\varphi_{F_1} \wedge \cdots \wedge \varphi_{F_n}$ by $\varphi_{\vec{F}}$ and identify a vector $\vec{c}$ with the tuple $(c_1, \ldots, c_n)$. Then, a composite system built from the specifications $c_i : \varphi_{F_i}$ can be given by a single pair $\vec{c} : \varphi_{\vec{F}}$. To say that a $[\varphi, \psi]$-style timing analysis T is correct and exact is nothing but the statement that $T[\varphi, \psi](\vec{F}, \vec{c})$ is the tightest-fitting stabilization bound for which the composite behaviour $[\![\vec{c} : \varphi_{\vec{F}}]\!]$ is well-timed w.r.t. the specification $\psi_{\vec{F}}$.

Definition 3.3 (Correctness and Exactness)
Let T be a $[\varphi, \psi]$-style timing analysis. T is *correct* and *exact* if for all lists of *fu*s $\vec{F}$, stabilisation bounds $\vec{c} = c_1, \ldots, c_n$ with $c_i \in |\varphi_{F_i}|$, and $d \in |\psi_{\vec{F}}|$ the following equivalence holds:

$$d \sqsupseteq T[\varphi, \psi](\vec{F}, \vec{c}) \quad \Leftrightarrow \quad [\![\vec{c} : \varphi_{\vec{F}}]\!] \subseteq [\![d : \psi_{\vec{F}}]\!]$$

where the inequation $d \sqsupseteq T[\varphi, \psi](\vec{F}, \vec{c})$ on the left implicitly includes the statement that $T[\varphi, \psi](\vec{F}, \vec{c})$ is defined.

If we view a timing analysis T as a formal derivability relation between combined temporal and functional specifications, then correctness and exactness of T can be rephrased as soundness and completeness conditions of logic. Formally, let us define $c_1 : \varphi_{F_1}, \ldots, c_n : \varphi_{F_n} \vdash_T d : \psi_{\vec{F}}$ as an abbreviation for the statement that $T[\varphi, \psi](\vec{F}, \vec{c})$ is defined and $T[\varphi, \psi](\vec{F}, \vec{c}) \sqsubseteq d$. The semantical aspect can by formalised by a model-theoretic entailment relation $c_1 : \varphi_{F_1}, \ldots, c_n : \varphi_{F_n} \models d : \psi_{\vec{F}}$ with the natural definition: $\forall C. (\forall i. C \models c_i : \varphi_{F_i}) \Rightarrow C \models d : \psi_{\vec{F}}$, which is the same as $[\![\vec{c} : \varphi_{\vec{F}}]\!] \subseteq [\![d : \psi_{\vec{F}}]\!]$. Then, correctness and exactness come down to the bi-implication

$$c_1 : \varphi_{F_1}, \ldots, c_n : \varphi_{F_n} \vdash_T d : \psi_{\vec{F}} \quad \Leftrightarrow \quad c_1 : \varphi_{F_1}, \ldots, c_n : \varphi_{F_n} \models d : \psi_{\vec{F}}$$

where $\Rightarrow$ is the soundness and $\Leftarrow$ the completeness direction relating syntactic and semantic notions of logical entailment. In general, of course, $\vdash_T$ may be quite different from syntactic derivability in a logic calculus, yet in some cases it may be just that (see *e.g.* [26]).

A correct $[\varphi, \psi]$-style timing analysis T, in general, will be a partial function since for some choices of $\vec{F}$ and $\vec{c}$ it may happen that $[\![\vec{c} : \varphi_{\vec{F}}]\!]$ cannot be well timed for $\psi_{\vec{F}}$. Such a situation, for instance, occurs in static delay analysis (see Sec. 3.1.6). Since we also want T to be computable this implies that T must effectively recognise if the composite behaviour cannot be well-timed for $\psi_{\vec{F}}$. Alternatively, we may insist on $T[\varphi, \psi]$ being total by making the specification mapping ψ clever enough to adjust $\psi_{\vec{F}}$, depending on $\vec{F}$, in such a way that $[\![\vec{c} : \varphi_{\vec{F}}]\!]$ is always well timed for $\psi_{\vec{F}}$, for all $\vec{c}$. In fact, the construction of $\psi_{\vec{F}}$ may well be part of the algorithm T. In this context it is important to note that a timing analysis, in general, not only computes the timing but also computes or verifies the function. How much of the function it verifies depends on the choice of ψ. In particular, all algorithms published in the literature that perform a data-dependent analysis must necessarily verify some amount of functional behaviour as well. Our notion of correctness (alias soundness) and exactness (alias completeness) just makes this explicit.

Soundness and completeness define a relationship between timing analysis algorithms and specification styles $[\varphi, \psi]$. The game can be played in two directions: Given an existing timing analysis algorithm X determine a pair $[\varphi, \psi]$ such that X is a correct and exact $[\varphi, \psi]$-style timing analysis. Then we may say that X is *characterised* by $[\varphi, \psi]$. The other direction is to start from a specification style $[\varphi, \psi]$ and try to find an algorithm T that is a correct and exact $[\varphi, \psi]$-style timing analysis. Then, T may be viewed as a *realisation* of a $[\varphi, \psi]$-style analysis. The following theorem is a direct consequence of Proposition 2.2:

Theorem 3.4 (Existence of Correct and Exact Timing Analyses)
For all choices $\varphi : X \mapsto \varphi_X$ $\psi : \vec{Y} \mapsto \psi_{\vec{Y}}$ of mappings, translating *fu*s into timing models (as specified in Definition 3.2) there exists a correct and exact $[\varphi, \psi]$-style timing analysis. This analysis $T[\varphi, \psi]$ is uniquely determined by φ and ψ.

Note that Theorem 3.4 only states the existence of correct and exact timing analyses characterised by timing models. It does not give any indication of how to construct one.

To relate different $[\varphi, \psi]$-style analyses we must study the relationship between timing models. There are several ways in which a timing model φ_1 can be related to a timing model φ_2. In this paper we adopt the simple *extensional* viewpoint which is to compare φ_1 and φ_2 in terms of the class of behaviours that can be well timed for them. Suppose we know that all C that can be well timed for φ_1 can also be well timed for φ_2. In other words, for all C, $C \models c : \varphi_1$ entails $C \models c : \varphi_2$. This implies (by axiom of choice) there exists a function $f : |\varphi_1| \to |\varphi_2|$ such that for all $c \in |\varphi_1|$, if $C \models c : \varphi_1$ then $C \models f(c) : \varphi_2$, or equivalently $C \models f : \varphi_1 \supset \varphi_2$. Such a function f, uniformly in C, translates stabilisation bounds for φ_1 into those for φ_2. To claim that such a function exists is equivalent to stipulating that the set of all waveforms $\mathbb{S} \to \mathbb{N} \to \mathbb{B}$ can be well timed for the implication $\varphi_1 \supset \varphi_2$, which in turn is the same as saying that $\varphi_1 \supset \varphi_2$ is a theorem of PST. In the same way, φ_1 and φ_2 share the same well timed behaviours *iff* the equivalence $\varphi_1 \equiv \varphi_2$ is a theorem of PST. In practice, for plugging together different timing analyses and to relate timing models, we will be interested not in arbitrary comparison functions but in exact ones, *i.e.* those that turn optimal bounds into optimal bounds and thus preserve the exactness of the analysis.

3.1 On the Variety of Timing Models

Now, we finally come to play the game. We characterise a range of $[\varphi, \psi]$-style timing analyses by varying φ and ψ. We will discuss 6 timing models, called tft (ternary function table), prm (prime cover), smp (simple worst case), tpl (topological), cls (classic), stt (static), which represent 6 different ways of systematically transforming a Boolean function into a PST specification. These models are only a few of the many possibilities, but indicate the different dimensions in which the granularity of the data-dependency of delays may be adjusted. When arranged according to their extensional semantics we get the following picture in which the abstractness

$$\begin{tikzcd}
\mathsf{tft} \arrow[r, leftrightarrow, "\equiv"'] & \mathsf{prm} \arrow[r, leftrightarrow, "\equiv"'] & \mathsf{smp} \arrow[r, "\supset"'] & \mathsf{tpl} \arrow[r, "\supset"] & \mathsf{cls} \\
 & & & \mathsf{stt} \arrow[ur, "\supset"'] &
\end{tikzcd}$$

of the models increases from left to right. The most discriminative is tft, which associates a stabilisation bound with every partial input state, the most abstract is cls which only specifies stationary behaviour. While the (extensional) equivalences $\equiv$ between tft, prm, smp, involve a loss of precision in the timing but not in function, the proper inclusions $\supset$ between smp, tpl, cls, stt also involve a loss of function, *i.e.* some transitions that are bounded in the model to the left of $\supset$ cannot be timed by the model on the right of $\supset$. Note that the static model stt is incomparable with all the models considered here except cls.

Using these timing models we obtain the following table characterising some of the published timing analyses for combinational circuits:

Analysis	Style	Source
Topological Delay	$[\mathsf{tpl}, \mathsf{tpl}]$	
Floating Mode Sensitization	$[\mathsf{smp}, \mathsf{tpl}]$	Chen/Du [4], Devadas *et al.* [7]
Viability Mode Sensitization	$[\mathsf{smp}, \mathsf{tpl}]$	McGeer/Brayton [23]
"A New Approach"	$[\mathsf{prm}, \mathsf{prm}]$	Huang/Parng/Shyu [29]
Static Sensitization	$[\mathsf{stt}, \mathsf{tpl}]$	Benkoski *et al.* [2]
Proof Extraction	$[\{\mathsf{tpl}, \mathsf{prm}\}, \{\mathsf{tpl}, \mathsf{prm}\}]$	Fairtlough/Mendler [26]

Note that floating and viability mode analysis are characterised by the same models, so they compute the same delay. This was proven already in [23]. As is seen most timing analyses are $T[\varphi, \mathsf{tpl}]$, so that the discriminative parameter for classification is the specification mapping φ pertaining to the components. It is typical for standard algorithms that for the components a data-dependent timing model is employed but for the circuit itself all information is collapsed into only a single worst-case topological delay. A more refined hierarchical method based on static sensitization has been proposed in [15], which we conjecture to be characterised as a $[\mathsf{stt}, \mathsf{stt}]$-style analysis.

In the following sub-sections we will discuss different timing models using the example of a complex gate, seen in Fig. 1, with Boolean function $d = (a \cdot b) + \bar{c}$.

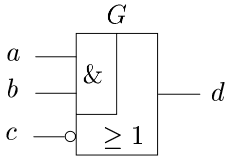

FIG. 1. A simple complex gate G

As a point of reference for our semantic discussions the three-valued function table of G seen in Fig. 2 will be useful.

a	0	$\frac{1}{2}$	1	0	$\frac{1}{2}$	1	0	$\frac{1}{2}$	1	0	$\frac{1}{2}$	1	0	$\frac{1}{2}$	1	0	$\frac{1}{2}$	1	0	$\frac{1}{2}$	1	0	$\frac{1}{2}$	1	0	$\frac{1}{2}$	1
b	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	1	1	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	1	1	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	1	1
c	0	0	0	0	0	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	1	1	1	1	1	1	1	1
d	1	1	1	1	1	1	1	1	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	0	0	0	0	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	1

FIG. 2. Three-valued function table of gate G

The function table in Fig. 2 is the three-valued extension [40, 11] of the Boolean equation $d = (a \cdot b) + \bar{c}$, specifying the stationary behaviour of the gate. In the following subsections we are going to specify this three-valued functional behaviour in various ways by PST formulas, which differ, essentially, in how much information we are representing in the truth values and how much in the signal values. The basic principle is, the richer the structure of the formula the richer the timing information that is captured. As indicated already, we do not intend to explore the range of possibilities in a systematic way, but focus on a few examples that link up with standard timing analyses.

3.1.1 Classical Delay-free Specification

We begin with the extreme case of purely functional behaviour. Recall from Section 2.3 that double negation $\neg\neg\varphi$ translates every PST formula φ into a classical statement about the stationary state of the signals mentioned in φ. Double negation eliminates all timing information that may be contained in φ. Within the scope of a double negation $\neg\neg\varphi$ the logical connectives $\wedge, \vee, \supset$ take their classical meaning, and the atomic proposition $s = v$ reads "s eventually stabilises to v." Thus, the delay-free functional behaviour can be expressed by a doubly negated formula. The following specification cls_G captures the ternary function table of the complex gate G (Fig. 2) as a relation between the stationary states of input signals a, b, c and output signal d:

$$\begin{aligned} \mathsf{cls}_G \quad \equiv_{df} \quad & \neg\neg(((\neg a = 0 \wedge \neg b = 0) \vee \neg c = 1 \vee d = 0) \\ & \wedge ((\neg c = 0 \wedge \neg a = 1) \vee (\neg c = 0 \wedge \neg b = 1) \vee d = 1)). \end{aligned}$$

Equivalently we may use the specification $\mathsf{cls}_G = \neg\neg(d = a \cdot b + \bar{c})$ with the abbreviations introduced in Section 2.4. We find $|\mathsf{cls}_G| \cong \underline{1}$, so cls_G does not contain any timing information. A circuit behaviour C is well timed for cls_G *iff* the stationary states $V^\infty \in \mathbb{S} \to \mathbb{K}$ assumed by the waveforms $V \in C$ are consistent with the ternary function table of G.

3.1.2 Complete Three-Valued Function Table

On the other end of the scale lies the specification of three-valued functional behaviour in which every possible ternary input pattern is represented by a separate transition *input-state* $\supset \bigcirc$ *output-state* with its own characteristic delay. For the complex gate G with its three input signals a, b, c we get 27 different three-valued input pattern, and thus a conjunction of 27 implications

$$\begin{aligned} \mathsf{tft}_G \quad \equiv_{df} \quad & (a = 0 \wedge b = 0 \wedge c = 0 \supset \bigcirc(d = 1)) \wedge \\ & (a = \frac{1}{2} \wedge b = 0 \wedge c = 0 \supset \bigcirc(d = 1)) \wedge \\ & (a = 1 \wedge b = 0 \wedge c = 0 \supset \bigcirc(d = 1)) \wedge \\ & \dots \\ & (a = 0 \wedge b = 1 \wedge c = 1 \supset \bigcirc(d = 0)) \wedge \\ & (a = \frac{1}{2} \wedge b = 1 \wedge c = 1 \supset \bigcirc(d = \frac{1}{2})) \wedge \\ & (a = 1 \wedge b = 1 \wedge c = 1 \supset \bigcirc(d = 1)), \end{aligned}$$

in which the n-th conjunct corresponds to the n-th column in the function table of Fig. 2. Using the equivalence $(s = \frac{1}{2}) \equiv$ *true* the formula tft_G, which is elementary, may be simplified in the obvious way, without loosing intensional timing information. One verifies that $\neg\neg\mathsf{tft}_G \equiv \mathsf{cls}_G$ is a theorem, and that $|\mathsf{tft}_G| \cong \mathbb{N}^{\underline{27}}$. Hence, tft_G is a timing model of G with stabilisation bounds (up to order isomorphism) being 27-tuples of delays.

The degree of data-dependency of delays in tft_G goes beyond what is handled by standard timing analysis algorithms. There, in many cases a single worst-case delay value is assumed for each primitive component and from this a single worst-case delay is derived for the composite circuit.

It has been observed that for simple CMOS gates, for instance, the propagation delay shows rather big variations depending on the input context and that the knowledge of these differences can be exploited for the construction of wave-pipelining circuits [14]. While for primitive gates we may refer to physics, the input data dependency is more obvious for composite circuits, on logic grounds, just as it is evident for software programs whose computation time depends on the input data. In practice, it will depend on the concrete implementation technology and the intended precision of the modelling in how far these distinctions regarding the delays of components or a composite system are necessary. The specification tft_G marks an extreme case that can be relaxed in various ways.

3.1.3 Delay by Coverings

In the range between the two extreme cases, cls_G with 0 and tft_G with 27 implicit timing parameters, intermediate variants can be found. Given an arbitrary covering of the ternary input space by subsets we can design a specification that associates with every subset of the cover a single delay, which is independent of the output values generated by the input pattern contained in the subset.

Rather than presenting the general method, we discuss a distinguished case for our example gate. We consider the canonical representation obtained by covering, for each output state separately (these are $d = 1$ and $d = 0$), the associated input conditions by all prime implicants. The resulting formula is

$$\begin{array}{lll}\mathsf{prm}_G & \equiv_{df} & ((a = 0 \wedge c = 1) \vee (b = 0 \wedge c = 1)) \supset \bigcirc(d = 0) \;\wedge \\ & & ((c = 0) \vee (a = 1 \wedge b = 1)) \supset \bigcirc(d = 1),\end{array}$$

where $a = 0 \wedge c = 1$ and $b = 0 \wedge c = 1$ are the two prime input cubes to cover all input conditions that produce output $d = 0$, and the two prime input cubes $c = 0$ and $a = 1 \wedge b = 1$ produce output $d = 1$. We find that $|\mathsf{prm}_G| \cong \mathbb{N}^4_\bot$, which means that prm_G implicitly distinguishes 4 delay values. We also have $\neg\neg\mathsf{prm}_G \equiv \mathsf{cls}_G$, so that prm_G specifies the same stationary behaviour as cls_G. Since prm_G is elementary it is timing model of G.

It can be shown that, extensionally, prm_G contains the same information as tft_G, *i.e.* $\mathsf{prm}_G \equiv \mathsf{tft}_G$, but that they are not intensionally equivalent. Intensionally, tft_G is more informative, since it contains a delay not only for all prime input cubes, as prm_G does, but for all input cubes that produce a definite output response. For instance, while prm_G only has one delay for the input pattern $a = 0 \wedge c = 1$ which works regardless of the behaviour of input b, tft_G also has delays for the more specific situations $a = 0 \wedge c = 1 \wedge b = 0$ and $a = 0 \wedge c = 1 \wedge b = 1$ in which input b is known to be stable as well. Therefore, intensionally, prm_G is a timing abstraction of tft_G in which some information is given up. Technically this can be formalised by a Galois connection $g \dashv f : \mathsf{tft}_G \equiv \mathsf{prm}_G$ on the worst-case stabilisation bounds of

tft_G and prm_G. This means there exists a pair of functions $g \in |\mathsf{tft}_G| \to |\mathsf{prm}_G|$ and $f \in |\mathsf{prm}_G| \to |\mathsf{tft}_G|$ with the following property: For all $C \subseteq \mathbb{S} \to \mathbb{N} \to \mathbb{B}$, $x \in |\mathsf{tft}_G|$, $y \in |\mathsf{prm}_G|$ such that x is worst-case for C and tft_G and y is worst-case for C and prm_G, we have $x \sqsubseteq f(y)$ *iff* $g(x) \sqsubseteq y$.

Taking specification by implicants prm_G both for components and composite circuit characterises the “New Approach” timing analysis presented in [29].

3.1.4 Simple Data Dependency

Many standard timing analyses employ a simple form of data dependency with a single worst-case delay value that applies for all ternary input pattern, but the circuit or component need not wait for all inputs to arrive before it produces an output. This is a special form of data-dependency in which it is not the value of the delay that depends on the input but its activation. In our example this simple data dependency of timing is specified by

$$\begin{aligned} \mathsf{smp}_G \quad &\equiv_{df} \quad \mathsf{activate}(a,b,c) \supset \bigcirc(d = (a \cdot b) + \bar{c}) \\ \mathsf{activate}(a,b,c) \quad &\equiv_{df} \quad (a = 0 \land c = 1) \oplus (b = 0 \land c = 1) \oplus (c = 0) \oplus (a = 1 \land b = 1). \end{aligned}$$

With smp_G we model a single transition of the form *activation* $\supset \bigcirc$ *output equation* the stabilisation bounds of which record a single delay. In fact, $|\mathsf{smp}_G| \cong \mathbb{N}$. The antecedent $\mathsf{activate}$ of the implication collects all input situations (waveforms) that activate the gate. We have $\neg\neg\mathsf{smp}_G \equiv \mathsf{cls}_G$ again, whence the elementrary smp_G is a timing model of G.

Notice that smp_G is quite similar to prm_G. However, instead of static choice $\vee$ for the input activation of prm_G we use dynamic choice $\oplus$ (*cf.* Sec. 2.5) in smp_G; Also, the gate's functionality, which resides in the structure of the formula prm_G, now in smp_G has been pushed into the signal values, *i.e.* the output equation $d = (a \cdot b) + \bar{c}$.

Let us expand a bit more on the relationship between prm_G and smp_G. It is not difficult to convince oneself that both specifications are extensionally equivalent, *i.e.* that $\mathsf{prm}_G \equiv \mathsf{smp}_G$ is theorem of PST. This implies that there are functions $f : |\mathsf{prm}_G| \to |\mathsf{smp}_G|$ and $g : |\mathsf{smp}_G| \to |\mathsf{prm}_G|$ which translate stabilisation bounds of prm_G into those of smp_G, and vice versa. It can be shown that among the possible pairs f, g there is no isomorphism, *i.e.* prm_G and smp_G are not intensionally equivalent for timing analysis. It turns out that the best solution for such comparison functions again is a Galois connection $f \dashv g : \mathsf{prm}_G \equiv \mathsf{smp}_G$. To be more precise, modulo the canonical order isomorphisms $|\mathsf{prm}_G| \cong \mathbb{N}^4$ and $|\mathsf{smp}_d| \cong \mathbb{N}$ this Galois connection is given by the duplication function $g = \lambda n.\,(n,n,n,n) : \mathbb{N} \to \mathbb{N}^4$ and the maximum $f = \lambda(n_1,n_2,n_3,n_4).max(n_1,n_2,n_3,n_4) : \mathbb{N}^4 \to \mathbb{N}$. But this is what we would expect. In order to compactify a description prm_G with four delays into a description smp_G with only one delay, we take the worst-case over all input situations that are still distinguished in prm_G. The other way round, nothing needs to be done. Since the delay contained in smp_G covers all input pattern it merely needs to be duplicated to give an upper bound for each of the four input pattern of prm_G.
It is clear that once we have abstracted from the four delays of prm_G to only one of smp_G, by taking maximum, we have lost timing information. From the single

worst-case delay we cannot recover the original distinctions. The Galois connection $f \dashv g : \mathsf{prm}_G \equiv \mathsf{smp}_G$ formalises this timing abstraction.

3.1.5 Topological Delay

As mentioned before it depends on the intention of the user how much of the intensional precision of PST is relevant. In the extreme case we may be content with a single worst-case delay under the worst-case assumption that the component requires all its inputs to be stable before it starts to compute the output. In case of our example this would be achieved by the following PST formula

$$\mathsf{tpl}_G \quad \equiv_{df} \quad \mathsf{const}(a) \wedge \mathsf{const}(b) \wedge \mathsf{const}(c) \supset \bigcirc(d = (a \cdot b) + \bar{c}).$$

This is quite similar to the formula smp_G for simple data dependency, in that it features a single input-output transition. However, now, the gate is considered activated only if all three inputs have become stable. We have $\neg\neg\mathsf{tpl}_G \equiv \mathsf{cls}_G$ and $|\mathsf{tpl}_G| \cong \mathbb{N}$.

Since $\mathsf{const}(a) \wedge \mathsf{const}(b) \wedge \mathsf{const}(c)$ implies $\mathsf{activate}(a, b, c)$, the implication $\mathsf{smp}_G \supset \mathsf{tpl}_G$ is a PST theorem. The other direction does not hold, as one can verify, so that tpl_G is a proper weakening of smp_G, *i.e.* encompasses a strictly larger set of waveforms. In every circuit in which G occurs a component we can replace the specification tpl_G of our example gate by the stronger smp_G, and still verify the same consequences for the waveforms of the composite circuit. However, since we have reduced the possible waveforms the resulting total worst-case delay, in general, will have become smaller. This is a model-theoretic way of saying that replacing the topological delay model by a less conservative, *i.e.* more exact, delay model, results in better approximations.

If we specify both the primitive gates of a circuit and the composite system according to this principle of topological delay, then the worst-case stabilisation bounds coincide with the *topological* delay, *i.e.* the length of the longest path through the circuit. If only the composite circuit is described by a worst-case formula tpl_G but the components specified with simple data dependency smp_G we characterise *viability* [23] or *floating-mode* analysis [4, 7].

3.1.6 Static Path Delay

One of the earliest data-dependent timing analyses is the so-called *static path analysis*, which is based on *static path sensitization* [2]. Here the propagation delay is determined by the longest path through a circuit that can be activated by a *controlling* signal transition on a single input, assuming that the signals on all side-inputs to the path have reached *non-controlling* stable values. Take a two-input AND gate, for instance. The non-controlling value for an input of the AND is 1 since then the output is uniquely determined by the value of the other input. This can be generalised accordingly to multiple-input gates. The following PST formula is the timing model for our example gate that captures this static sensitization mode of operation:

$$\begin{aligned}
\mathsf{stt}_G \quad \overset{\text{def}}{=} \quad & ((\mathsf{const}(a) \wedge \neg\neg(c = 1 \wedge b = 1)) \supset \bigcirc(d = a)) \wedge \\
& \quad ((\mathsf{const}(b) \wedge \neg\neg(c = 1 \wedge a = 1)) \supset \bigcirc(d = b)) \wedge \\
& \qquad ((\mathsf{const}(c) \wedge \neg\neg(a = 0 \vee b = 0)) \supset \bigcirc(d = \bar{c})) \wedge
\end{aligned}$$

$$\neg\neg((c = 0 \vee (a = 1 \wedge b = 1)) \supset d = 1).$$

The first three conjuncts contain $\bigcirc$ and hence timing information. They state that there are three input conditions $\mathsf{const}(a) \wedge \neg\neg(c = 1 \wedge b = 1)$, $\mathsf{const}(b) \wedge \neg\neg(c = 1 \wedge a = 1)$, and $\mathsf{const}(c) \wedge \neg\neg(a = 0 \vee b = 0)$ for which the output d produces a response in bounded time. In each of them exactly one input signal controls the output, while for the remaining side-inputs the non-controlling stationary values are assumed. For instance, in $\mathsf{const}(a) \wedge \neg\neg(c = 1 \wedge b = 1)$ the controlling input is a while b, c are the side-inputs. The controlling input a is required to be stable $\mathsf{const}(a)$ while the double negation $\neg\neg(c = 1 \wedge b = 1)$ only refers to the final stationary state of the side-inputs. The stationary state $c = 1 \wedge b = 1$ is such that the output of the gate is uniquely determined by the controlling input a, *i.e.* we have the response $d = a$.

It is not difficult to see that the first three conjuncts of stt_G do not force any definite stationary behaviour for the input combination $a = 1 \wedge b = 1$ when nothing is known about c, and for $c = 0$ when nothing is known about a, b. In all of these cases the output is $d = 1$. These 5 columns of the ternary function table (Fig. 2) are missing in the transitions and thus have not assigned any delay. In order to ensure again that the stationary behaviour $\neg\neg\mathsf{stt}_G$ implied by stt_G is equivalent to cls_G, these 5 entries must be covered by a separate fourth conjunct $\neg\neg((c = 0 \vee (a = 1 \wedge b = 1)) \supset d = 1)$ of stt_G.

With the fourth conjunct the static timing model stt_G is complete for the stationary behaviour. As regards transition behaviour, however, it is still incomplete. This is a well-known feature of the static model. In our example it is not possible to verify the valid transition $c = 0 \supset \bigcirc(d = 1)$ from stt_G. It is not a semantic consequence of stt_G although it does occur in the intended concrete level behaviour of G. This means that if we use stt_G to specify our gate as a component of a larger circuit, then we may not be able to verify, and thus derive stabilisation delays for those transitions of the composite system that pass through the $c = 0$ input of the gate G. But if then, as usual, we take as the delay of the composite circuit the maximal delay over all verifiable transitions we may underestimate the true delay of the composite system. This is a new and simple way to explain why static path sensitization is not an exact criterion [34]. Since standard algorithms (in particular for static sensitization) make no attempt to specify the semantics of an analysis, missing transitions are not detected. Here is where PST as a specification language for combined functional and timing analysis pays off: The logic formula stt_G separates in a very precise way the part of the functional behaviour that is included in the delay information (first three conjuncts) and the part that is not (last conjunct). Wherever we use stt_G to analyse a composite system we will be told by the semantics (or by a sound and complete theorem prover, or a correct and exact timing analysis) that certain transitions cannot be verified for the composite system, and hence no delay can be computed. This may be unproblematic if the environment in which the circuit is to be used does not need to rely on this functionality, *e.g.* if it has a “don’t care” behaviour. Without a rigorous specification formalism such as PST these “don’t care” situations cannot be exploited, nor can the compositionality principle by which a complete analysis may be obtained from combining several incomplete analyses. This is the main methodological benefit of using a logic framework such as PST. In particular, the fact that we keep track of the coverage of the analysis can be used to patch the exactness problem for static sensitization analysis.

Note that $|\mathsf{stt}_G| \cong \mathbb{N}^3$, whence the timing model stt_G contains three delays. It is not difficult to find a formulation with only one delay, by using the dynamic choice operator $\varphi \oplus \psi$. Also, it should be clear that besides stt_G which corresponds to static path analysis as considered in the literature, there are other variants of PST specification styles which are incomplete w.r.t. delay information. These may be more or less complete, and more or less interesting, but surely there are many possibilities to vary the theme, here.

4 Some Meta-Theoretic Results about PST

This section sums up a few general results concerning the internal structure of PST and expressiveness issues. Though our analysis will be rather brief we hope to include enough material to justify our interest in PST. Our aim is to show that PST deserves to be studied in its own right as an intuitionistic modal theory, independently of the application to timing analysis that we put forward in this paper.

4.1 The Intuitionistic Nature of PST

Let us start off with a few basic observations. Using the properties of the realisability interpretation it is easy to see that PST is closed under modus ponens, *i.e.* if $\varphi \in$ PST and $\varphi \supset \psi \in$ PST then $\psi \in$ PST. It can be shown that PST properly extends PLL. The inclusion PLL $\subseteq$ PST follows from the fact that PST satisfies all axioms of intuitionistic logic, the three modal axioms $\bigcirc I, \bigcirc M, \bigcirc S$ of PLL, and the rule $\varphi \supset \psi \in \text{PST} \Rightarrow \bigcirc\varphi \supset \bigcirc\psi \in \text{PST}$. The inclusion is proper since, *e.g*, $\neg\bigcirc \mathit{false} \in$ PST but $\neg\bigcirc \mathit{false} \notin$ PLL (see [12]). Further, from the realisability semantics one obtains immediately that PST has the disjunction property and, like PLL, satisfies the inverse rule of necessitation.

PROPOSITION 4.1
(i) $\varphi \vee \psi \in$ PST implies $\varphi \in$ PST or $\psi \in$ PST
(ii) $\bigcirc\varphi \in$ PST implies $\varphi \in$ PST.

The disjunction property (i) implies that PST is a constructive theory. The rule (ii) essentially means that the modality $\bigcirc$ is redundant as a top-level operator. To see this note that since $\varphi \supset \bigcirc\varphi \in$ PST (the axiom $\bigcirc I$ of PLL) (ii) implies that $\varphi \in$ PST *iff* $\bigcirc\varphi \in$ PST. The semantics of $\bigcirc$ resides in the interplay with the other operators, notably implication. In fact, $\bigcirc$ turns out to be essentially intuitionistic in character. It is incompatible with classical principles. If such a principle is added then $\bigcirc$ trivialises in the sense that $\bigcirc\varphi \equiv \varphi$ becomes derivable. Semantically it is clear why this must be the case. A behaviour C satisfies the classical axiom $\neg\neg a = 0 \supset a = 0$ *iff* for every waveform $V \in C$ it is the case that if signal a stabilises to 0 eventually, then it must be stable already at time 0. Thus, by adding to PST the classical principles $\neg\neg a = 0 \supset a = 0$ and $\neg\neg a = 1 \supset a = 1$ for all $a \in \mathbb{S}$ we are essentially saying that all signal are stationary in all $V \in C$, *i.e.* the stabilization behaviour does not change in time. This means that the truth of a formula does not change with time either, whence there is no difference between eventual truth expressed by $\neg\neg\varphi$ and bounded truth modelled by $\bigcirc\varphi$.

Proposition 4.2
$\mathrm{PST} + \neg\neg\varphi \supset \varphi \vdash \bigcirc\varphi \equiv \varphi$

Proposition 4.2 is proven by using the fact that $\bigcirc\varphi \supset \neg\neg\varphi \in \mathrm{PST}$. This implies that $\bigcirc\varphi \supset \varphi$ by composition with the classical principle $\neg\neg\varphi \supset \varphi$. Since also $\varphi \supset \bigcirc\varphi \in \mathrm{PST}$ we finally derive $\bigcirc\varphi \equiv \varphi$.

4.2 On the Algebraic Structure of PST

Let $[\varphi]$ be the equivalence class of φ relative to PST, *i.e.* the set of formulas ψ such that $\varphi \equiv \psi \in \mathrm{PST}$. These equivalence classes together with the partial ordering $[\varphi] \leq [\psi]$ *iff* $\varphi \supset \psi \in \mathrm{PST}$ form a relatively pseudocomplemented distributive lattice, *i.e.* a Heyting algebra $(\mathrm{PST}, \leq)$. The properties of the modality $\bigcirc$ make it a modal operator on this Heyting algebra. A modal operator [19] on a $\wedge$-semi lattice $(H, \leq)$ is a mapping $j : H \rightarrow H$ that is inflationary $x \leq j(x)$, idempotent $jj(x) = j(x)$, and $\wedge$-preserving $j(x \wedge y) = j(x) \wedge j(y)$. We can give a simple description of the structure of the sub-algebra $\mathrm{PST}(a)$ of PST generated by a single fixed signal name $a \in \mathbb{S}$.

First we introduce the notion of a *constraint frame*. Constraint frames induce modal Heyting algebras just as Kripke frames induce Heyting algebras. They provide an adequate Kripke style semantics for PLL [12]. Below we will show that $\mathrm{PST}(a)$ is the intuitionistic theory of a very specific constraint frame.

Definition 4.3
A *constraint frame* is a structure (W, R_i, R_m, F) where W is a set, R_i, R_m are two partial orderings on W such that R_m is a sub-relation of R_i, and F is a subset of W that is upper closed with respect to R_i (and thus also for R_m).

In a constraint frame W is a set of Kripke worlds, and R_i and R_m are two accessibility relations used to interpret the intuitionistic implication $\supset$ and the modality $\bigcirc$, respectively. The last component F represents a set of fallible worlds which are the denotation of *false*. The reader is referred to [12] for more information on using constraint frames for Kripke models of PLL. The class of constraint frames relevant here for our deconstructing of $\mathrm{PST}(a)$ are the initial intervals $\mathbf{n} = \{0, 1, 2, \ldots, n-1\}$ of natural numbers with R_i being the natural ordering $\leq$, $R_m = \{\, (k, k+1) \mid k+1 < n \;\; \& \;\; k \text{ odd}\,\} \cup \{\, (k,k) \mid k < n\,\}$, and $F = \{n-1\}$. By the cartesian product $W_1 \times W_2$ of two constraint frames $(W_1, R_{i1}, R_{m1}, F_1)$ and $(W_2, R_{i2}, R_{m2}, F_2)$ we mean the constraint frame $(W_1 \times W_2, R_i, R_m, F_1 \times F_2)$ in which all operations are taken component-wise. Thus, $(w_1, w_2) \preceq (v_1, v_2)$ *iff* $w_1 \preceq_1 v_1 \;\&\; w_2 \preceq_2 v_2$ with $\preceq$ being R_i or R_m, respectively.

Proposition 4.4
Every constraint frame (W, R_i, R_m, F) induces a modal Heyting algebra $(\Upsilon_{R_i,F,W}, \subseteq, j_{R_m})$ where

- $\Upsilon_{R_i,F,W}$ is the set of upper closed subsets of the partial ordering $(W \setminus F, R_i)$,
- $(\Upsilon_{R_i,F,W}, \subseteq)$ is the Heyting algebra structure induced on $\Upsilon_{R_i,F,W}$ in the standard way by set inclusion; specifically, implication $X \supset Y$ is the set $\bigcup\{\, Z \in \Upsilon_{R_i,F,W} \mid Z \cap X \subseteq Y\,\}$.
- $j_{R_m} : \Upsilon_{R_i,F,W} \rightarrow \Upsilon_{R_i,F,W}$ is the modal operator defined by $j_{R_m}(X) := \{\, w \in W \setminus F \mid \forall v.\, wR_iv \Rightarrow \exists u.\, vR_mu \;\&\; u \in F \cup X\,\}$.

It is not difficult to see that $j_{R_m}(X)$ is an upper closed subset of $W \setminus F$. Also, one verifies that j_{R_m} satisfies the equation $X \supset j_{R_m}(Y) = j_{R_m}(X) \supset j_{R_m}(Y)$, which by Theorem 1.3 of [19] suffices to make j_{R_m} a modal operator.

THEOREM 4.5
The modal Heyting algebra $(\mathrm{PST}(a), \wedge, \vee, \supset, \mathit{false}, \mathit{true}, \bigcirc)$ generated by a single fixed $a \in \mathbb{S}$ is isomorphic to the modal Heyting algebra induced by the constraint frame $\mathbf{4} \times \mathbf{4} \times \mathbf{2}$.

Theorem 4.5 implies that $\mathrm{PST}(a)$ is a finite algebra. Contrast this with intuitionistic logic for which the Lindenbaum algebra in one atomic proposition, known as the Rieger-Nishimura lattice, is infinite (see *e.g.* [38]).

4.3 Relationship with Intermediate Logics of Kreisel-Putnam, Dummett, and Medvedev

Let us take a closer look at the intuitionistic base, *i.e.* the $\bigcirc$-free fragment of PST. We call this modal-free fragment PST-I. We show that PST-I is related to two well-known intuitionistic intermediate logics, *viz.* the logic KP of Kreisel-Putnam [16], Dummett's linear logic LC [10] and Medvedev's intermediate logic MV of finite problems [24]. Building on the results of [27] a new characterization of MV is derived.

In comparing with intermediate logics we must be careful to bear in mind that the atomic propositions $a = 1$, $a = 0$ in PST are propositional constants rather than variables. This means that the formulas of PST-I are not schematic, whereas formulas in intermediate logics are schematic in propositional variables. To stress this distinction we will refer to PST-I as a theory and to its elements as propositions and to KP, LC, and MV as logics and call their elements formulas. To be more precise, let us define a *theory* to be a collection of propositions of intuitionistic logic in propositional constants $a = 0, a = 1$ $(a \in \mathbb{S})$ that is closed under Modus Ponens. A *logic* is a collection of formulas in propositional variables, say $\alpha, \beta, \ldots$, that is closed under Modus Ponens *and* Substitution. Every theory T induces a logic, written $S(T)$ and called the *standard part* or the *standardization* of T: $S(T)$ is the largest set of formulas φ such that all propositional instantiations of φ are contained in T. In other words, $S(T)$ is the collection of axiom schemes that are valid in T.

With these preliminaries we can return to our program of relating PST-I to other constructive logics. First of all, one can show that the standard part $S(\text{PST-I})$ strictly extends intuitionistic propositional logic IPC but is properly included in classical propositional logic CPC. As to strictness we note that $S(\text{PST-I})$ contains the Kreisel-Putnam axiom scheme (see *e.g.* [38]) $(\neg\varphi \supset (\psi_1 \vee \psi_2)) \supset ((\neg\varphi \supset \psi_1) \vee (\neg\varphi \supset \psi_2))$ which is not a theorem of IPC. At the other end $S(\text{PST-I})$ refutes the CPC axiom of the Excluded Middle $\varphi \vee \neg\varphi$. Thus, $\mathrm{IPC} \subsetneq \mathrm{KP} \subseteq S(\text{PST-I}) \subsetneq \mathrm{CPC}$, where KP is the logic obtained from extending IPC with the Kreisel-Putnam scheme. This means that $S(\text{PST-I})$ is an intermediate (also called superintuitionistic) logic. Its position in the lattice of intermediate logics will be highlighted in the following. We show that the intermediate logics of Medvedev and of Dummett coincide with the standard part of two special theories of PST-I, and furthermore that MV coincides with the standard part of PST-I itself.

Consider the intermediate logic LC of Dummett [10]. It can be defined as the intuitionistic theory of the ω-chain, *i.e.* the frame $(\mathbb{N}, \leq)$ of natural numbers under the natural ordering. It is not difficult to see that every intuitionistic model built on this frame can be simulated by a behaviour $C = \{V^\delta \mid \delta \in \mathbb{N}\}$ generated from a single waveform $V : \mathbb{S} \to \mathbb{N} \to \mathbb{B}$. We may call these the *linear* behaviours. Now, let PST-IL $\supset$ PST-I be the intuitionistic theory of linear behaviours, *i.e.*

$$\text{PST-IL} \quad := \quad \{\varphi \mid \varphi \text{ is } \bigcirc\text{-free and } C \models \varphi \text{ for all linear } C\}.$$

It turns out that the subclass of linear behaviours can be characterised by the linearity axiom $(\varphi \supset \psi) \vee (\psi \supset \varphi)$, which also completely axiomatises LC [10]. In fact, PST-IL can be characterised formally as the class of propositions that are derivable from PST-I by adding the axiom scheme $[(\varphi \supset \psi) \vee (\psi \supset \varphi)]$ (*i.e.* all $\bigcirc$-free propositional instances) and closing under Modus Ponens. This is the content of the following

Proposition 4.6
PST-IL = PST-I + $[(\varphi \supset \psi) \vee (\psi \supset \varphi)]$.

We now have the following relationship between LC and PST-IL:

Proposition 4.7
S(PST-IL) = LC.

Next, we come to Medvedev's intermediate logic MV of finite problems [24], which, too, is closely related to PST-I. We first show that S(PST-I) $\subseteq$ MV. To this end we consider the theory of another special subclass of behaviours, *viz.* the constant behaviours. A behaviour $C \subseteq \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ is *constant* if for all $V \in C$ and all $a \in \mathbb{S}$, $\forall t.\, V(a)(t) = 0$ or $\forall t.\, V(a)(t) = 1$, *i.e.* if all signals in every waveform are constant 0 or 1. Consider the (proper) extension

$$\text{PST-IC} \quad := \quad \{\varphi \mid \varphi \text{ is } \bigcirc\text{-free and } C \models \varphi \text{ for all constant } C\}$$

of PST-I. Again, there is an equivalent axiomatic definition, in terms of the axioms

$$\mathsf{const}(a) \quad = \quad (\neg\neg a = 0 \supset a = 0) \wedge (\neg\neg a = 1 \supset a = 1) \wedge \neg\neg(a = 0 \vee a = 1)$$

expressing that signal a is constant (*cf.* Example 2.9):

Proposition 4.8
PST-IC = PST-I + $[\mathsf{const}(a)]$.

The important feature of constant behaviours C is that the time dimension of the model is removed. Every waveform $V \in C$ can be identified with a Boolean valuation $V \in \mathbb{S} \to \mathbb{B}$ such that $V(a) = 0$ if signal a is constant 0 and $V(a) = 1$ if a is constant 1. The realisability semantics for atoms then simplifies to $V \models 0 : a = v$ *iff* $V(a) = v$. Moreover, for all $\delta \in \mathbb{N}$ we have $V^\delta = V$. This implies that the realisability clause for implication $V \models f : \varphi \supset \psi$ simplifies to $\forall x \in |\varphi|.\, V \models x : \varphi \Rightarrow V \models fx : \psi$. This is precisely the classical set-theoretic realisability interpretation of Medvedev [24]. Taking account of our special convention $|a = 0| = |a = 1| = \underline{1}$ which associates a singleton set of realisers with every atom we conclude that PST-IC coincides with Medvedev's theory of singleton problems. This theory is termed $\mathrm{F_{cl}}$ in [27], *i.e.* PST-IC = $\mathrm{F_{cl}}$. By Theorem 11 of [27], MV = $S(\mathrm{F_{cl}})$, whence we get

Proposition 4.9
$S(\text{PST-IC}) = \text{MV}$.

From the inclusion PST-I $\subseteq$ PST-IC, the monotonicity of the standardisation operator, and from Proposition 4.9 we finally obtain $S(\text{PST-I}) \subseteq \text{MV}$ as claimed. Now, the other direction can be shown to hold, too, so that we have:

Theorem 4.10
$S(\text{PST-I}) = \text{MV}$.

The proof is roughly this: In Theorem 18 of [27] is shown that $\text{MV} = S(\text{F}_{\text{int}})$, where F_{int} is the theory of "intuitionistic singleton problems." This theory is defined like PST-I except that it is based on arbitrary Kripke models rather than waveforms as is PST-I. Since waveforms are just linear Kripke models we get $S(\text{F}_{\text{int}}) \subseteq S(\text{PST-I})$. This, then, implies $\text{MV} \subseteq S(\text{PST-I})$, as desired.

Theorem 4.10 means that $S(\text{PST-I})$ cannot be finitely axiomatized by purely structural schemes since MV cannot [20]. However, this does not exclude that PST-I itself is finitely axiomatizable by non-structural axioms. Indeed, we conjecture that PST-I can be axiomatized by the axiom schemes

$$\begin{array}{l}(\zeta \supset (\varphi \vee \psi)) \supset ((\zeta \supset \varphi) \vee (\zeta \supset \psi))\\(((\varphi \supset \psi) \supset \zeta) \wedge ((\psi \supset \varphi) \supset \zeta)) \supset \zeta\\\neg(a = 1 \wedge a = 0),\end{array}$$

where ζ ranges over $\{\vee, \bigcirc\}$-free formulas. The first of the three axiom schemes is a variant of KP, it reflects the set structure of behaviours. The second is a specialisation of the linearity axiom of LC. Indeed, if we would allow arbitrary instantiations for ζ, then the second axiom is interderivable with $(\varphi \supset \psi) \vee (\psi \supset \varphi)$. This second axiom reflects the linear nature of waveforms. The third scheme $\neg(a = 1 \wedge a = 0)$ is due to the special interpretation of our atoms.

4.4 On Expressiveness

Since, after all, PST is a propositional theory one may wonder just how much timing behaviour can be expressed and how this compares with more conventional logic specification formalisms. Not much to be expected, on the face of it. Nevertheless, the answer to this question turns out to be nontrivial. Some results on expressiveness are presented here, the exact characterisation still remains open. In as much as PST expresses the difference between bounded and unbounded stabilisation it is stronger than more conventional but less specialised languages such as classical propositional temporal and modal logics, first-order predicate logic, or Büchi's monadic second-order logic over one successor. Recall that the PST formula $a = 1 \supset \bigcirc(b = 1)$ says that "*whenever a stabilises to* 1 *then after a bounded response time b goes* 1 *as well.*" Such fixed but unknown stabilisation delay, which is tantamount to generic timing analysis, cannot be expressed in these other formalisms by closed formulas, in particular not without introducing free time parameters. On the other hand, as far as transient behaviour is concerned PST is considerably weaker than the mentioned classical formalisms. Our semantics of propositions must satisfy some rigid intuitionistic closure properties which restrict expressibility in PST rather drastically:

Proposition 4.11
PST is (i) time invariant, *i.e.* $C \models \varphi \Rightarrow C^{\delta} \models \varphi$, and (ii) closed under sub-behaviours, *i.e.* $C \models \varphi \;\;\&\;\; D \subseteq C \Rightarrow D \models \varphi$.

In view of Proposition 4.11 the reader is reminded, however, that PST is not proposed to substitute general purpose logics. Its virtues stem from being a special purpose theory to capture stabilisation behaviour for finite combinational systems. For such applications we are interested only in whether or not a signal a has stabilised at a given time rather than the precise changing of a over time. This is reflected by the fact that with atomic proposition $a = 0$, $a = 1$ we cannot access the value of a at any particular time like we can with atomic propositions of temporal or predicate logics. So, for instance, we cannot capture transient behaviours, such as "*signal a switches exactly* 5 *times before it stabilises,*" or "*the distance between two changes of a is at most* 100 *time units.*"

So, what can be expressed, then? Let us analyse the different kinds of stabilisation behaviours that can be distinguished for a fixed given signal $a \in \mathbb{S}$. To begin with there are the three basic options which relate to *constant* $a = 0, a = 1$, *bounded* $\bigcirc(a = 0), \bigcirc(a = 1)$, and *stationary* $\neg\neg a = 0, \neg\neg a = 1$ stabilisation modes. We know from Theorem 4.5 that the fragment PST(a) corresponds to the modal Heyting algebra generated by the upper closed subsets of the constraint frame $\mathbf{4} \times \mathbf{4} \times \mathbf{2}$. This is a finite but certainly rich lattice of stabilisation properties. As far as expressiveness is concerned this internal algebraic characterization, still, is not very useful. A much better idea of PST(a) as a specification formalism for classes of behaviours is obtained from the following characterization in terms of second-order classical predicate logic:

Theorem 4.12
PST(a) captures precisely all properties of waveform sets $C \subseteq \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ expressible in classical second-order predicate logic by closed formulas ξ in the language of the primitive stabilisation predicate $V(a) \downarrow_t b$ with waveform variable V, value variable b, time variable t, subject to the following restriction: In the prenex normal form of ξ

- all occurrences of waveform variables are universally quantified
- every negative occurrence of a time variable is universally quantified
- no universally quantified time variable occurs both positively and negatively.

In Theorem 4.12 a positive (negative) occurrence of a variable in ξ means an occurrence that is in the scope of an even (odd) number of negations. The semantics $C \models_c \xi$ of a closed formula ξ is given as in classical predicate logic with the understanding that every universal quantification $\forall V$ implicitly quantifies over the set C, *i.e.* is read as $\forall V \in C$. Let us call the closed formulas of second-order predicate logic with the syntactic restrictions given in the Theorem *stabilization sentences.* The theorem can be proven by a systematic analysis and suitable normalisation of stabilization sentences in second-order predicate logic on the one side and the propositions φ in PST(a) on the other. For every stabilization sentence ξ in normal form one constructs a PST(a) proposition φ_ξ such that $C \models_c \xi$ *iff* $C \models \varphi_\xi$. Vice versa, for every PST(a) proposition φ in normal form one constructs a stabilization sentence ξ_φ such that $C \models \varphi$ *iff* $C \models_c \xi_\varphi$.

Example 4.13
For an indication of how the two formalisms relate pick the following simple class of stabilization sentences. Consider the formulas in prenex normal form over the atomic matrix $V(a) \downarrow_t b$, *i.e.* formulas $Q_1. Q_2. Q_3. V(a) \downarrow_t b$ in which $Q_1. Q_3. Q_3$ quantify the three variables V, b, t such that the restrictions of Theorem 4.12 are fulfilled. Table 1 lists all possible quantifications and the corresponding $\mathrm{PST}(a)$ proposition expressing the same stabilization behaviour. Notice how the different choices and orderings of quantifiers of predicate logic are captured merely by propositional means using the intuitionistic semantics in $\mathrm{PST}(a)$. Note also that all propositional connectives $\wedge, \vee, \supset, \bigcirc, \neg$, which are all independent in PST, are involved in expressing the different quantification schemes.

Predicate Logic	$\mathrm{PST}(a)$
$\forall V. \forall t. \forall b. V(a) \downarrow_t b$	*false*
$\forall V. \exists t. \exists b. V(a) \downarrow_t b$	$\neg\neg(a = 0 \vee a = 1)$
$\exists b. \forall V. \exists t. V(a) \downarrow_t b$	$\neg\neg a = 0 \vee \neg\neg a = 1$
$\exists t. \forall V. \exists b. V(a) \downarrow_t b$	$\neg\neg(a = 0 \vee a = 1) \wedge \bigcirc(\neg\neg a = 0 \supset a = 0 \wedge \neg\neg a = 1 \supset a = 1)$
$\exists t. \exists b. \forall V. V(a) \downarrow_t b$	$\bigcirc(a = 0 \vee a = 1)$
$\forall V. \forall t. \exists b. V(a) \downarrow_t b$	$\neg\neg(a = 0 \vee a = 1) \wedge \neg\neg a = 0 \supset a = 0 \wedge \neg\neg a = 1 \supset a = 1$
$\exists b. \forall V. \forall t. V(a) \downarrow_t b$	$a = 0 \vee a = 1$

Table 1. Predicate Logic and $\mathrm{PST}(a)$

Theorem 4.12 provides a rather satisfactory characterisation of expressiveness of single signal propositions. A complete characterisation of the expressive power of full PST is still open. It can be shown that in the $\{\vee, \bigcirc\}$-free fragment of PST arbitrary orderings for the stabilization of signals can be specified. In order to make this more precise, suppose, from now on, we are interested only in the stabilization of a finite set of signals $S \subset \mathbb{S}$. Let $\mathbb{A} = \{a = 0, a = 1 \mid a \in S\}$ be the set of atoms over these signals. By a *state* we mean a subset $\sigma \subseteq \mathbb{A}$ of atoms such that for no $a \in S$ both $a = 0, a = 1 \in \sigma$. The state *assumed* by a waveform $V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ at time t is the set $\sigma_t V := \{a = v \in \mathbb{A} \mid V(a) \downarrow_t v\}$. The set $\sigma V = \{\sigma_t V \mid t \in \mathbb{N}\}$ is linearly ordered under subset inclusion and called the *stabilization sequence*, or simply *s-sequence*, of V. It captures the sequence and relative ordering in which all signals from S stabilise (or not) in V, but abstracts from the absolute occurrence time and absolute distances of stabilization events. Every linearly ordered subset of states may occur as the s-sequence of some waveform. The following theorem says that the s-sequences of all waveforms in a behaviour can be specified uniquely by $\{\vee, \bigcirc\}$-free propositions.

Theorem 4.14
For every subset Σ of s-sequences (over signals S) there exists a $\{\vee, \bigcirc\}$-free proposition φ_Σ such that $C \models \varphi_\Sigma$ *iff* for all $V \in C$, $\sigma V \in \Sigma$.

Example 4.15
To give an example, the formula $(a = 1 \equiv b = 1) \wedge (c = 0 \supset a = 1) \wedge \neg\neg(a = 1 \wedge b = 1 \wedge c = 0)$ says "*a and b eventually stabilise to* 1 *simultaneously, whereupon, but not earlier, c stabilises at* 0." This corresponds to the two possible s-sequences

$\Sigma = \{\sigma_1, \sigma_2\}$ where $\sigma_1 = \{\{a = 1, b = 1\}, \{a = 1, b = 1, c = 1\}\}$ and $\sigma_2 = \{\{a = 1, b = 1, c = 1\}\}$.

5 Conclusion

The paper presented a new specification language PST to capture the stabilisation behaviour of finite combinational systems. Its semantics being sets of Boolean waveforms PST combines both the temporal with the functional aspects. Yet, in contrast to conventional logic formalisms it does not intermingle the two at the syntactic level. The syntax is purely propositional with an additional modal operator $\bigcirc$ that acts as a generic place-holder for stabilisation bounds. The bounds themselves are treated as realisers of propositions and a defining ingredient of the intuitionistic semantics.

The relevance of PST as a unifying framework lies in the fact that it allows us to specify and compare different timing analyses in terms of their underlying timing models. One and the same Boolean function can be represented in many ways as PST formula, giving rise to various different timing models that associate different stabilisation delays with different parts of the functionality. We have characterised some published algorithms as correct and complete PST-style analyses. In setting up a timing model we can play with two parameters. One is the granularity of the data-dependency of the delay. It can be varied in large limits, distinguishing different sets of input conditions with different delays. These input conditions may determine the activation of a computation and the value of the delay separately. The value may also depend on how “strongly” the circuit is activated by the input. The second parameter we can play with is the amount of functionality that is included in the delay analysis. In general, a PST timing model specifies delays only for a relevant part of the input space and output behaviour, explicitly including “don’t care” or “don’t know” situations. This is important to make rigorous sense of incomplete timing analyses such as static sensitization. All these different timing models can be related in PST extensionally by logic implication $\supset$ and equivalence $\equiv$, measuring the classes of circuit behaviours that can be well timed for them, and intensionally by giving explicit comparison functions translating stabilisation bounds between the models. Galois connections specify intensional timing abstraction and timing approximations induced by passing from one model to another.

Being able to handle different timing models with varying degree of data-dependency within one framework suggests PST as a distinguished formalism for hierarchical timing analysis. It is evident that if we are to construct the timing of a large circuit in a compositional way, then for efficiency reasons we cannot maintain the same (high) degree of timing granularity all the way up through the hierarchy. To handle a complex sub-circuit we must lump together many input states into a single activation pattern, for which only one worst-case delay is recorded, and only keep distinctions where this involves significant differences in the associated delays. Also, the information about data must be compressed to give input-output relations, or nondeterministic functions. All this can be done in PST. Specifically, nondeterministic pattern $s_1, \ldots, s_n \in E[s_1, \ldots, s_n]$ can be encoded, which state that the stable value of the signal vector $(s_1, \ldots, s_n)$ is in the set described by the (ternary) expression E. Then,

$$a_1, \ldots, a_n \in E[a_1, \ldots, a_n] \quad \supset \quad \bigcirc(b_1, \ldots, b_m \in F[b_1, \ldots, b_m])$$

may specify an input-output transition of some component up to data abstraction: "*if the input $\vec{a}$ stabilises in the set E, then with bounded response time the output $\vec{b}$ stabilises in the set F.*" In this way quite abstract descriptions can be produced.

We believe that the inherent compositionality of PST is a major advantage over algebraic formalisms, in particular Timed Boolean Functions (TBFs) [17] which have recently been proposed as a unifying model to specify and implement timing analyses both for combinational as well as synchronous circuits. However, since TBFs are descriptions of arbitrary waveforms (though only with a finite number of signal changes) they can capture transient behaviour, and thus are more expressive than PST. To compare the amount of information handled by the two formalisms, roughly, we get the following picture: TBFs describe the Boolean relationship between the stable values of signals in a number of contiguous intervals

$$(-\infty, t_0), (t_0, t_1), \dots, (t_{n-1}, t_n), (t_n, +\infty),$$

whereas in PST we relate only the stable values of signals in their final stabilisation intervals $(t_n, +\infty)$. Thus, TBFs are able to specify timing analyses for dynamic sensitization, or two-vector delay models [17, 18] which is not possible in PST.

It is not surprising that timing analyses based on the more "accurate" dynamic modelling of TBFs result in smaller delay times. However, like in the two-vector model, these are computed for quite specific input conditions, which may not necessarily be guaranteed by the environment in which the circuit is used. In particular, feedback loops cannot be handled in a satisfactory way. In PST no such structural or behavioural assumptions are imposed on the environment, and thus it can be used as well to analyse asynchronous combinational systems such as the ones considered in [21]. Also, the TBF model does not support abstraction and refinement of timing models, which requires a certain degree of nondeterminism, or looseness, in specifications. TBFs are deterministic functions from input waveforms to output waveforms with fixed input timing. PST specifications, in contrast, are relational and do not fix the timing parameters. In PST we can compose the timing models of components into a more abstract timing model of the composite circuit, and thus reduce information without loosing correctness. Dynamic analysis based on TBFs also suffers from the so-called *monotone speedup failure* [23]: Reducing the delays of circuit components may increase the worst-case delay computed for a composite circuit on the basis of a TBF model.

On the theoretical side it would be interesting to characterise fully the expressiveness of PST and to explore systematically the lattice of possible timing models for a given Boolean function. On the practical side, it seems an intriguing idea to devise a generic timing analysis algorithm based on theorem proving for PST. We envisage an algorithm T, which, given any list $c_1 : \varphi_1, \dots, c_n : \varphi_n$ of "calibrated" timing models φ_i and a circuit specification ψ, computes a worst-case stabilisation bound $c \in |\psi|$ such that $c_1 : \varphi_1, \dots, c_n : \varphi_n \vdash_T c : \psi$. Such an algorithm would encompass all $[\varphi, \psi]$-style analyses together. It might be obtained from an intensionally sound and complete proof system for PST, or at least the fragment of elementary propositions. We conjecture that such a proof system exists. Note that although decidability of intuitionistic propositional logic is P-SPACE complete, theorem proving for the specialised PST theory of elementary propositions of PST need not be less efficient than standard timing analysis algorithms. The complexity of the decision problem for PST proper and the existence of a complete axiomatization for it is another issue that needs to be addressed in future work.

Acknowledgements
I am grateful to Matt Fairtlough and Pierangelo Miglioli for many helpful discussions. Thanks go to Gerald Lüttgen and Robert Kießling for useful comments on a preliminary version of this paper. I would also like to thank the anonymous referees for their constructive criticisms and suggestions which helped me to improve the paper. This work was supported by the *Deutsche Forschungsgemeinschaft* under grant number Me 1427/1-3.

References

[1] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity.* John Wiley and Sons, 1992.

[2] J. Benkoski, E. Vanden Meersch, L. J. M. Claesen, and H. De Man. Timing verification using statically sensitizable paths. *IEEE Transactions on Computer-Aided Design*, 9(10):1073–1084, October 1990.

[3] N. Benton, G. Bierman, and V. de Paiva. Computational types from a logical perspective I. Technical Report, Computer Laboratory University of Cambridge, U.K., August 1993.

[4] H. C. Chen and D. H. Du. Path sensitization in critical path problem. In *ACM Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 1990.

[5] H.-C. Chen and D. H. C. Du. Path sensitization in critical path problem. In *International Conference on Computer-Aided Design*, pages 208–211, 1991.

[6] H. B. Curry. The elimination theorem when modality is present. *Journal of Symbolic Logic*, 17:249–265, 1952.

[7] S. Devadas, K. Keutzer, and S. Malik. Delay computation in combinational logic circuits: Theory and algorithms. In *International Conference on Computer-Aided Design*, pages 176–179, 1991.

[8] S. Devadas, K. Keutzer, and S. Malik. Certified timing verification and transition delay of a logic circuit. In *Proc. of the 29th Design Automation Conference*, June 1992.

[9] D. H. C. Du, S. H. C. Yen, and S. Ghanta. On the general false path problem in timing analysis. In *Design Automation Conference*, pages 555–560, 1989.

[10] M. Dummett. A propositional calculus with a denumerable matrix. *Journal on Symbolic Logic*, 24:96–107, 1959.

[11] E. B. Eichelberger. Hazard detection in combinational and sequential switching circuits. *IBM J. Res. Div.*, 9:90–99, 1965.

[12] M. Fairtlough and M. Mendler. Propositional Lax Logic. *Information and Computation*, 137(1):1–33, August 1997.

[13] R. I. Goldblatt. Grothendieck topology as geometric modality. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 27:495–529, 1981.

[14] C. Th. Gray, W. Liu, and R. K. Cavin III. Optimal clocking for wave pipelined operation in multistage systems with feedback. In *International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, Malente, September 1993. ACM/GMD.

[15] P. Johannes, L. Claesen, and H. De Man. On the use of reconvergence analysis for efficient hierarchical static sensitizable path analysis. In *International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, Malente, September 1993. ACM/GMD.

[16] G. Kreisel and H. Putnam. Eine Unableitbarkeitsbeweismethode für den Intuitionistischen Aussagenkalkül. *Archiv für mathematische Logik und Grundlagenforschung*, pages 74–78, 1957.

[17] K. C. Lam and R. K. Brayton. *Timed Boolean Functions. A Unified Formalism for Exact Timing Analysis.* Kluwer, 1994.

[18] W. K. C. Lam, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Circuit delay models and their exact computation using timed boolean functions. In *Design Automation Conference*, pages 128–134, 1993.

[19] D. S. Macnab. Modal operators on Heyting algebras. *Algebra Universalis*, 12:5–29, 1981.

[20] L. L. Maksimova, D. P. Skvorcov, and V. B. Šehtman. The impossibility of a finite axiomatization of medvedev's logic of finitary problems. *Soviet Math Doklady*, 20(2):394–398, 1979.

[21] S. Malik. Analysis of cyclic combinational circuits. In *International Conference on Computer-Aided Design*, pages 618–625. IEEE, 1993.

[22] P. McGeer and R. Brayton. Efficient algorithms to computing the longest viable path in a combinational network. In *Design Automation Conference*, pages 561–567, June 1989.

[23] P. McGeer and R. Brayton. Provably correct critical paths. In *Proc. of the Decennial Caltech VLSI Conference*, 1989.

[24] Ju. T. Medvedev. Interpretation of logical formulas by means of finite problems. *Soviet Math. Dokl.*, 7(4):857–860, 1966.

[25] M. Mendler. Timing analysis of combinational circuits in intuitionistic propositional logic. *Formal Methods in System Design*, to appear 1999. A short preliminary version was presented at TABLEAUX'96, Springer, LNAI 1071, pp. 261–277.

[26] M. Mendler and M. Fairtlough. Ternary simulation: A refinement of binary functions or an abstraction of real-time behaviour? In M. Sheeran and S. Singh, editors, *Proceedings of the 3rd Workshop on Designing Correct Circuits (DCC96).* Springer, October 1996. Springer Electronic Workshops in Computing.

[27] P. Miglioli, U. Moscato, M. Ornaghi, S. Quazza, and G. Usberti. Some results on intermediate constructive logics. *Notre Dame Journal of Formal Logic*, 30(4):543–562, 1989.

[28] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[29] S.-T. Huang T.-M. Parng and J.-M. Shyu. A new approach to solving the false path problem in timing analysis. In *International Conference on Computer-Aided Design*, pages 216–219, 1991.

[30] S. Perremans, L. Claesen, and H. De Man. Static timing analysis of dynamically sensitizable paths. In *Design Automation Conference*, pages 568–573, June 1989.

[31] W. Rautenberg. *Klassische und Nichtklassische Aussagenlogik.* Vieweg und Sohn, Braunschweig/Wiesbaden, 1979.

[32] J. P. Silva, K. A. Sakallah, and L. M. Vidigal. FPD – an environment for exact timing analysis. In *International Conference on Computer-Aided Design*, pages 212–215, 1991.

[33] J. P. M. Marques Silva and K. A. Sakallah. An analysis of path sensitization criteria. In *Proc. ICCD*, pages 68–72, 1993.

[34] J. P. Marques Silva and K. A. Sakallah. A comparison of path sensitization criteria for timing analysis. In *International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, Malente, September 1993. ACM/GMD.

[35] J. P. Marques Silva and K. A. Sakallah. Sensitization networks for accurate timing analysis. In *International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, Malente, September 1993. ACM/GMD.

[36] R. Stewart and J. Benkoski. Static timing analysis using interval constraints. In *International Conference on Computer-Aided Design*, pages 308–311, 1991.

[37] A. S. Troelstra. Realizability. In S. R. Buss, editor, *Handbook of Proof Theory*, chapter VI, pages 407–474. Elsevier, 1998.

[38] D. van Dalen. Intuitionistic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume III, chapter 4, pages 225–339. Reidel, 1986.

[39] L. Verdoscia and R. Vaccaro. A high-level dataflow system. *Computing*, 60:285–305, 1998.

[40] M. Yoeli and S. Rinon. Application of ternary algebra to the study of static hazards. *Journal of the ACM*, 11:84–97, 1964.

Received 30 September 1998.