A Timing Refinement of Intuitionistic Proofs and its Application to the Timing Analysis of Combinational Circuits

Michael Mendler

Abstract

Up until now classical logic has been the logic of choice in formal hardware verification. This report advances the application of intuitionistic logic to the timing analysis of digital circuits. The intuitionistic setting serves two purposes at the same time. The model-theoretic properties are exploited to handle the second-order nature of bounded delays in a purely propositional setting without need to introduce explicit time and temporal operators. The proof theoretic properties are exploited to extract quantitative timing information and to reintroduce explicit time in a convenient and systematic way.

We present a natural Kripke-style semantics for intuitionistic propositional logic, as a special case of a Kripke *constraint model* for Propositional Lax Logic [4], in which validity is validity *up to stabilization.* We show that this semantics is equivalently characterized in terms of *stabilization bounds* so that implication $\supset$ comes out as "*boundedly gives rise to.*" An intensional semantics for proofs is presented which allows us effectively to compute quantitative stabilization bounds.

We discuss the application of the theory to the timing analysis of combinational circuits. To test our ideas we have implemented an experimental prototype tool and run several simple examples.

1 Motivation

A recurring theme in Computer Science, as an engineering discipline, is to find the right level of abstraction in modeling real-world phenomena. There are as many levels of abstraction as there are modelling applications and to adjust the level properly in each case is a nontrivial undertaking. On the one hand it must not be too abstract so as to compromise the correctness of the model by ignoring important low-level features. At the same time it must not be too concrete so as to clutter up the model by irrelevant low-level details.

A typical case where this issue arises is the formal reasoning about digital circuits in the presence of timing constraints. The ideal abstract specification of a combinational block computing the function f, for instance, is the equation

$$y = f(x), \tag{1}$$

where x stands for the input and y for the output of the circuit. Usually, x and y would be Boolean vectors and f a Boolean function over x. Though this algebraic specification certainly is adequate for a lot of situations, it is but an approximation that completely elides the timing dimension of the real circuit, which cannot be ignored in practice. When we establish the correctness of a synchronous circuit, for instance, knowledge of the stabilization behaviour of the combinational blocks is crucial to adjust the clock rate and to meet the setup and hold constraints of the register flipflops. In this case a more refined description of the combinational block is much more adequate, like

$$\forall t.\ \forall a.\ (\forall \tau \geq t.\ x(\tau) = a) \ \Rightarrow\ (\forall \tau \geq t + 100ns.\ y(\tau) = f(a)). \tag{2}$$

It says that whenever the input x stabilizes at some value a then after a delay of $100ns$ the output y stabilizes at the value $f(a)$. This is a safe description of a typical (functional) circuit representing the idealized behaviour (1) with a stabilization imprecision of $100ns$. In choosing this level of description we are guaranteed to stay in control of not only the functional but also the stabilization behaviour. For instance, when this circuit is now composed with another one realizing $z = g(y)$ subject to an imprecision of $23ns$ (in the above sense) we can formally derive a composite behaviour $z = g(f(x))$ subject to an imprecision of $123ns$:

$$\begin{aligned}
&\forall t.\ \forall a.\ \mathit{stable}(x, t, a) \ \Rightarrow\ \mathit{stable}(y, t + 100ns, f(a)) \ \wedge \\
&\quad \forall t.\ \forall a.\ \mathit{stable}(y, t, a) \ \Rightarrow\ \mathit{stable}(z, t + 23ns, g(a)) \\
&\Rightarrow\ \forall t.\ \forall a.\ \mathit{stable}(x, t, a) \ \Rightarrow\ \mathit{stable}(z, 123ns, g(f(a))),
\end{aligned}$$

where $\mathit{stable}(x, t, a)$ abbreviates $\forall \tau \geq t.\ x(\tau) = a$. From an engineering point of view, however, such a detailed low-level analysis, though it can be done, is not necessarily a good idea. It mixes up what usually are considered separate concerns: the abstract reasoning in terms of time-less algebraic equations, *i.e.* the inference of $z = g(f(x))$ from $y = f(x)$ and $z = g(y)$, and the analysis of the timing constraint, *i.e.* adding the delays $100ns + 23ns = 123ns$.

The purpose of this report is to present a meet-in-the-middle framework for the formal verification of combinational circuits which lies between the two extremes (1) and (2). It is not as concrete as (2) which messes up the functional verification with timing, and it is not as abstract as (1) which looses track of the stabilization behaviour altogether.

2 Introduction

Let us agree, for the purpose of this report, that we are interested only in Boolean combinational circuits, *i.e.* circuits built from components like Inv, And, Or, Nand gates, *etc.* An extension to combinational circuits over arbitrary finite data domains is straightforward. A *signal* a, then, is a timed boolean function, *i.e.* $a \in \mathbb{N} \to \mathbb{B}$, where time is represented by the natural numbers. For convenience we will fix a countably infinite number of signals $\mathbb{S} = \{a, b, c, c_1, c_2, \cdots\}$ throughout, and conceive a circuit as a relation $C \subseteq \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ which constrains the behaviour on (a finite number of) signals. The elements $V \in C$ will be called *observable behaviours*, *executions*, or *valuations*.

For every $a \in \mathbb{S}$ let $a = 1$ and $a = 0$ be the atomic sentences stating that signal a is stable at 1 and 0, respectively. Thus, with $C \models \phi$ expressing that circuit C satisfies formula ϕ, we stipulate

$$\begin{aligned} C \models a = 1 \quad &=_{df} \quad \forall V \in C.\, \mathit{stable}(V(a), 0, 1) \\ C \models a = 0 \quad &=_{df} \quad \forall V \in C.\, \mathit{stable}(V(a), 0, 0), \end{aligned}$$

where $\mathit{stable}(x, t, b)$, as before, abbreviates $\forall \tau \geq t.\, x(\tau) = b$. Our goal is to build up from these basic sentences more complex propositional formulas that capture the stabilization behaviour of circuits, without resorting to explicit time and explicit temporal operators. But how do we get nontrivial transition behaviour from purely propositional connectives? After all, the basic sentences represent constant behaviour, whence a simple combination of such sentences by ordinary logical disjunction, conjunction, and implication does not yield anything exciting. The trick is to interpret an implication like $a = 1 \supset b = 1$ not as in classical logic "if a is stable 1 then b is stable 1" but as a *gives rise to* statement: "there exists a stabilization bound δ so that whenever a becomes stable 1, b will become stable 1 with a maximal delay δ." Formally,

$$\begin{aligned} C \models a = 1 \supset b = 1 \quad =_{df} \quad & \exists \delta \in \mathbb{N}.\, \forall V \in C.\, \forall t \in \mathbb{N}. \\ & \mathit{stable}(V(a), t, 1) \Rightarrow \mathit{stable}(V(b), t + \delta, 1). \end{aligned} \tag{3}$$

Notice that this is a stronger condition than

$$\forall V \in C.\, \forall t \in \mathbb{N}.\, \mathit{stable}(V(a), t, 1) \Rightarrow \exists \delta \in \mathbb{N}.\, \mathit{stable}(V(b), t + \delta, 1), \tag{4}$$

which might seem a more obvious generalization of (2) to abstract from a particular stabilization constraint, such as $100ns$. The difference between (4) and (3) is a swapping of quantifiers, which is crucial: (4) allows C to exhibit unbounded response time, which is not what we want. In contrast, (3) specifies an unknown, but fixed stabilization delay for all observable behaviours of the circuit, which is the right generalization of (2). To put it another way, a propagation delay is not a property of an individual execution of a circuit, but of the circuit itself.

But how can we squeeze an intrinsically second-order condition into a single propositional connective like $\supset$ and yet hope to get a useful logic? Not much we can expect from this, it appears. Surprisingly, it turns out that the particular semantic condition (3) indeed behaves like an implication, *viz.* an intuitionistic one. In order to indicate why, let us first observe that (3) is closed under inclusion and time shift of C:

$$C \models a = 1 \supset b = 1 \quad \Rightarrow \quad \forall D \subseteq C.\, D \models a = 1 \supset b = 1 \;\;\&\;\; \forall \delta.\, C^\delta \models a = 1 \supset b = 1,$$

where C^δ is obtained from C by shifting all $V \in C$ left by an amount of δ. Formally, we define V^δ such that $V^\delta(a)(t) = V(a)(t + \delta)$, and then $C^\delta = \{\, V^\delta \mid V \in C \,\}$. This closure property reveals the intuitionistic nature of our interpretation. In fact, assuming that C is time invariant, *i.e.* $C^\delta \subseteq C$ for all $\delta \in \mathbb{N}$, it is possible to show (*cf.* Thm. 5.1) that $C \models a = 1 \supset b = 1$ is equivalent to

$$\forall D \subseteq C.\, D \models a = 1 \quad \Rightarrow \quad \exists \delta.\, D^\delta \models b = 1,$$

where D ranges over time invariant subsets. This formulation has the structure essentially of an intuitionistic implication. In a sense that can be made precise, it is the intuitionistic strengthening of (4). The corresponding Kripke frame is the set of time-invariant subsets $C \subseteq \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ ordered by inverse subset inclusion. In this way we will obtain a full-blooded intuitionistic semantics in which non-trivial stabilization behaviour can be expressed and verified.

3 Intuitionistic Propositional Logic LJ

We will be concerned with a non-standard semantics for standard intuitionistic propositional logic, the formulas of which are generated by the grammar

$$\phi \quad ::= \quad a = 1 \;\mid\; a = 0 \;\mid\; \mathit{false} \;\mid\; \phi \wedge \phi \;\mid\; \phi \vee \phi \;\mid\; \phi \supset \phi$$

where a ranges over the set of signal constants $\mathbb{S}$. We will use $\equiv$ to abbreviate bi-implication, and $\neg\phi$ to abbreviate $\phi \supset \mathit{false}$. The formulas $a = 1$, $a = 0$ are the atomic propositional sentences of the logic. Our formal calculus is a slight variant [3] of Gentzen's sequent calculus for the logic LJ [6]. The rules are shown in Fig. 1. A multi-succedent version of this calculus also appears in [2].

$$\frac{}{\Gamma, \phi \;\vdash\; \phi}\; id \qquad\qquad \frac{\Gamma, \Gamma', \phi \vdash \theta}{\Gamma, \phi, \Gamma' \vdash \theta}\; get$$

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi}\; \wedge R \qquad\qquad \frac{\Gamma, \phi, \psi \vdash \theta}{\Gamma, \phi \wedge \psi \vdash \theta}\; \wedge L$$

$$\frac{\Gamma, \phi \vdash \theta \qquad \Gamma, \psi \vdash \theta}{\Gamma, \phi \vee \psi \vdash \theta}\; \vee L \qquad\qquad \frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \vee \psi}\; \vee R_1 \qquad\qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \phi \vee \psi}\; \vee R_2$$

$$\frac{}{\Gamma, \mathit{false} \vdash \theta}\; \mathit{falseL}$$

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \supset \psi}\; \supset R \qquad\qquad \frac{\Gamma, \phi \supset \psi \vdash \phi \qquad \Gamma, \psi \vdash \theta}{\Gamma, \phi \supset \psi \vdash \theta}\; \supset L$$

Figure 1: Gentzen Rules for LJ.

The system of Fig. 1 is a cut and contraction free presentation of LJ that directly lends itself to be mechanized in a Prolog fashion. The only difficulty lies in the fact that, in a blind application of the rules in a backwards search, the replication of the principal formula $\phi \supset \psi$ in the left branch of rule $\supset L$ may lead to looping. This problem can be overcome by some form of loop detection, or more elegantly by using an equivalent system in which rule $\supset L$ is split up into four rules without replication [3].

For our purposes it will be important to have some notation for proofs and derivations. Derivation terms are linear notations for derivation trees in LJ with multiple conclusions. They are generated by the following simple grammar:

$$\begin{array}{rcl} \mathit{Deduction} & ::= & \mathit{Rule} \cdot \mathit{Deduction} \mid (\mathit{Deduction}, \mathit{Deduction}) \\ \mathit{Rule} & ::= & \mathit{id} \mid \mathit{get}(i) \mid \wedge R \mid \wedge L \mid \vee L \mid \vee R_1 \mid \vee R_2 \mid \mathit{falseL} \mid \supset R \mid \supset L, \end{array}$$

where i is a natural number recording the principal formula picked out by an application of rule *get*. If D is a derivation of a sequent $\Gamma \vdash \phi$ we will denote this by $D : \Gamma \vdash \phi$. When Γ is empty we call D a *proof* of ϕ.

4 Circuits as Timed Kripke Models

Hinging on the atomic sentences $a = 1$ and $a = 0$ we will interpret formulas of LJ as statements about circuits. Let ϕ be a formula and C a circuit. We say that C *validates* ϕ *up to stabilization*, written $C \models \phi$, if

- $\phi = \mathit{false}$ and $C = \emptyset$, or
- $\phi = a = 1$ and for all $V \in C$ and $t \in \mathbb{N}$, $V(a)(t) = 1$, or
- $\phi = a = 0$ and for all $V \in C$ and $t \in \mathbb{N}$, $V(a)(t) = 0$, or
- $\phi = \psi_1 \wedge \psi_2$ and both $C \models \psi_1$ and $C \models \psi_2$, or
- $\phi = \psi_1 \vee \psi_2$ and $C \models \psi_1$ or $C \models \psi_2$, or
- $\phi = \psi_1 \supset \psi_2$ and for all $D \subseteq C$ such that D is time invariant and $D \models \psi_1$, there exists $\delta \in \mathbb{N}$ such that $D^\delta \models \psi_2$.

A formula ϕ is *valid up to stabilization*, written $\models \phi$, if $C \models \phi$ for all circuits C.

Theorem 4.1 (Soundness I) *If* $\vdash \phi$ *then* $\models \phi$.

Proof: The theorem can be proven by induction on the derivation of $\vdash \phi$. It directly follows also from Theorems 5.1 and 6.4. ∎

Our definition of $\models$ nails down the Kripke semantics sketched in the Introduction. In this semantics $\phi \supset \psi$ is not the classical "if ϕ then ψ" but "ϕ gives rise to ψ in bounded time," where the stress is on *bounded*. We will elaborate on the connection between this Kripke semantics and bounded stabilization in the next section.

Notice that if $C = \emptyset$, then $C \models \phi$ for all ϕ. If C only consists of a single constant valuation V validity up to stabilization coincides with ordinary classical validity: $C \models \phi$ iff ϕ is classically valid for V, where an atomic sentence $a = i$, is read as "signal a is constant i." This special case corresponds to the usual static two-valued model of circuits. There is yet another way in which the classical two-valued model can be embedded into this semantics: As one verifies readily, $C \models \neg\neg\phi$ iff ϕ is classically valid on all $V \in C$, where $a = i$ is read as "signal a will stabilize to i." This means that double negated formulas are classical statements about the stationary state of a circuit. To be more precise, these are statements in a classical three-valued setting in which a signal a can be stable 1, stable 0, or oscillate. The latter value is represented by the formula $\neg a = 1 \wedge \neg a = 0$. If C is a circuit in which all signals eventually stabilize, then we get back, under double negation, the classical two-valued model of the final stable state.

Our claim that $\models$ is a Kripke semantics for LJ, of course, deserves some justification. It is not immediately obvious since the existential quantification $\exists\delta$ is not part of the usual Kripke interpretation of implication. However, it is possible to show that validity up to stabilization can be rephrased in terms of a *two-frame* intuitionistic Kripke semantics. More precisely, every circuit C induces a canonical Kripke *constraint model* [4] C^* such that for all formulas ϕ, $C \models \phi$ iff $C^* \models^* \phi^*$, where ϕ^* is obtained from ϕ by replacing each occurrence of a sub-formula $\psi \supset \theta$ by $\psi \supset \bigcirc\theta$. Here, $\bigcirc$ is the constraint modality and $\models^*$ validity of Propositional Lax Logic (PLL) [4]. In PLL the formula $\phi \supset \bigcirc\psi$ is interpreted as "if ϕ then under some constraint ψ." The interpretation of $\supset$ in this report is the special case in which "under some constraint" means "up to bounded propagation delay."

5 Stabilization Bounds

Let us unfold the meaning of the formula $a = 1 \supset b = 1$ for a circuit C. By definition $C \models a = 1 \supset b = 1$ is equivalent to

$$\forall D \subseteq C.\ (\forall V \in D.\ \mathit{stable}(V(a), 0, 1)) \Rightarrow (\exists\delta.\ \forall V \in D.\ \mathit{stable}(V(b), \delta, 1)) \qquad (5)$$

where the D tacitly is understood to range over time invariant subsets only. In particular, consider the time invariant subset $D^* = \{ V \in C \mid \mathit{stable}(V(a), 0, 1) \}$, which is defined so that $\forall V \in D^*.\ \mathit{stable}(V(a), 0, 1)$ is trivially true. If we instantiate D in (5) by D^*, then (5) reduces to $\exists\delta.\ \forall V \in D^*.\ \mathit{stable}(V(b), \delta, 1)$, which implies

$$\exists\delta.\ \forall V \in C. \forall t \in \mathbb{N}.\ \mathit{stable}(V(a), t, 1) \Rightarrow \mathit{stable}(V(b), t + \delta, 1). \qquad (6)$$

This is precisely the semantical condition (3) that we are aiming at. The converse can be shown, too, *viz.* that (6) implies (5). We can call δ a stabilization bound for the formula $a = 1 \supset b = 1$ and (6) the stabilization refinement of $a = 1 \supset b = 1$ by δ. This interpretation will now be lifted to all formulas.

To every formula ϕ we assign a set $[\![\phi]\!]$ representing, loosely speaking, the set of stabilization bounds for ϕ:

$$\begin{array}{rcl} [\![a = 1]\!] & = & \{*\} \;=\; [\![a = 0]\!] \\ [\![false]\!] & = & \{*\} \\ [\![\phi \wedge \psi]\!] & = & [\![\phi]\!] \times [\![\psi]\!] \\ [\![\phi \vee \psi]\!] & = & [\![\phi]\!] + [\![\psi]\!] \\ [\![\phi \supset \psi]\!] & = & [\![\phi]\!] \to \mathbb{N} \times [\![\psi]\!]. \end{array}$$

As usual the elements of the disjoint sum $[\![\phi]\!] + [\![\psi]\!]$ are pairs $(1, c_1)$ where $c_1 \in \phi$ or $(2, c_2)$ where $c_2 \in \psi$. Note that $[\![\phi]\!]$ always is non-empty, so that every formula has at least one bound. Let us call an element $c \in [\![\phi]\!]$ a *stabilization bound* or simply a *bound* for ϕ. The bounds can be used to refine formulas into dynamic statements in the time domain. We say that ϕ is *valid* with bound $c \in [\![\phi]\!]$ for a valuation $V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B}$, written $c, V \models_b \phi$, according to the following inductive clauses:

$$\begin{array}{rlll}
*, V & \models_b & a = 1 & \text{iff} \quad \text{for all } t \in \mathbb{N},\ V(a)(t) = 1 \\
*, V & \models_b & a = 0 & \text{iff} \quad \text{for all } t \in \mathbb{N},\ V(a)(t) = 0 \\
(c_1, c_2), V & \models_b & \phi \wedge \psi & \text{iff} \quad c_1, V \models_b \phi \text{ and } c_2, V \models_b \psi \\
(1, c_1), V & \models_b & \phi \vee \psi & \text{iff} \quad c_1, V \models_b \phi \\
(2, c_2), V & \models_b & \phi \vee \psi & \text{iff} \quad c_2, V \models_b \psi \\
f, V & \models_b & \phi \supset \psi & \text{iff} \quad \text{for all } t \in \mathbb{N} \text{ and } c \in [\![\phi]\!], \\
 & & & \qquad \text{if } c, V^t \models_b \phi \text{ then } \pi_2(f\,c), V^{\pi_1(f\,c)+t} \models_b \psi.
\end{array}$$

A formula ϕ is said to be *valid for a circuit C* with stabilization bound $c \in [\![\phi]\!]$, if $c, V \models_b \phi$ for all valuations $V \in C$. We call c a *uniform bound* for ϕ if ϕ is valid with bound c for all circuits C. Note that there are formulas ϕ, such as *false*, that do not have any uniform bound, even though $[\![\phi]\!]$ is nonempty. It can be checked easily by induction on ϕ, that if $c, V \models_b \phi$ then $c, V^t \models_b \phi$ for all $t \in \mathbb{N}$.
The following theorem implies that a formula ϕ is valid up to stabilization iff it has a uniform bound.

Theorem 5.1 (Completeness) *Let C be a circuit and ϕ a formula. Then C validates ϕ up to stabilization iff there exists a stabilization bound $c \in [\![\phi]\!]$ such that ϕ is valid for C with bound c.*

Proof: The proof proceeds by induction on ϕ.

• $C \models \mathit{false}$ is true iff $C = \emptyset$. Since $[\![\mathit{false}]\!] = \{*\}$ and $*, V \not\models_b \mathit{false}$ this is equivalent to $\exists c \in [\![\mathit{false}]\!].\ \forall V \in C.\ c, V \models_b \mathit{false}$.

• $C \models a = i$ is the condition $\forall V \in C.\ \forall t \in \mathbb{N}.\ V(a)(t) = i$. But this is equivalent to $\forall V \in C.\ *, V \models_b a = i$, which is equivalent to $\exists c \in [\![a = i]\!].\ \forall V \in C.\ *, V \models_b a = i$ as desired.

• Suppose $C \models \phi \wedge \psi$, *i.e.* $C \models \phi$ and $C \models \psi$. By induction hypothesis this implies the existence of $c_1 \in [\![\phi]\!]$ and $c_2 \in [\![\psi]\!]$ such that for all $V \in C$, $c_1, V \models_b \phi$ and $c_2, V \models_b \psi$. Thus, by definition $(c_1, c_2), V \models_b \phi \wedge \psi$. Since V is arbitrary and $(c_1, c_2) \in [\![\phi \wedge \psi]\!] = [\![\phi]\!] \times [\![\psi]\!]$ we are done.

Vice versa, suppose $c \in [\![\phi]\!] \times [\![\psi]\!]$ such that for all $V \in C$, $c, V \models_b \phi \wedge \psi$. Then, $c = (c_1, c_2)$ for some $c_1 \in [\![\phi]\!]$ and $c_2 \in [\![\psi]\!]$ and further, $c_1, V \models_b \phi$ and $c_2, V \models_b \psi$. Thus, by induction hypothesis, $C \models \phi$ and $C \models \psi$, whence $C \models \phi \wedge \psi$.

• Suppose $C \models \phi \vee \psi$, *i.e.* $C \models \phi$ or $C \models \psi$. Let us assume the first is true. Then, by induction hypothesis, there is $c \in [\![\phi]\!]$ such that for all $V \in C$, $c, V \models_b \phi$. This implies $(1, c), V \models_b \phi \vee \psi$ for all $V \in C$, which is what we need. The case that $C \models \psi$ is completely symmetric, replace 1 by 2.

Vice versa, suppose $c \in [\![\phi \vee \psi]\!] = [\![\phi]\!] + [\![\psi]\!]$ and $c, V \models_b \phi \vee \psi$ for all $V \in C$. If $c = (1, c_1)$ this means $c_1, V \models_b \phi$ for all $V \in C$, so that by induction hypothesis, $C \models \phi$. Symmetrically, if $c = (2, c_2)$ we conclude $C \models \psi$.

• Suppose there exists a function $f \in [\![\phi \supset \psi]\!] = [\![\phi]\!] \to \mathbb{N} \times [\![\psi]\!]$ such that for all $V \in C$, $f, V \models \phi \supset \psi$. By definition of $\models_b$ this means that for all $t \in \mathbb{N}$ and $c \in [\![\phi]\!]$ if $c, V^t \models_b \phi$ then $\pi_2\,(f\,c), V^{t+\pi_1\,(f\,c)} \models_b \psi$. We wish to show that $C \models \phi \supset \psi$. To this end let D be an arbitrary time invariant subset of C such that $D \models \phi$. By induction hypothesis, then, we have $c \in [\![\phi]\!]$ so that for all $V \in D$, $c, V \models_b \phi$. Using the property of function f we conclude that for all $V \in D$, $\pi_2\,(f\,c), V^{\pi_1\,(f\,c)} \models_b \psi$. But this is the same as saying that for all $V \in D^{\pi_1\,(f\,c)}$, $\pi_2\,(f\,c), V \models_b \psi$, which by induction hypothesis means $D^{\pi_1\,(f\,c)} \models \psi$. Hence, we have found, for all time invariant $D \subseteq C$ with $D \models \phi$, a $\delta \in \mathbb{N}$ such that $D^\delta \models \psi$. This, by definition, shows $C \models \phi \supset \psi$.

Vice versa, assume that $C \models \phi \supset \psi$, *i.e.* for all time invariant $D \subseteq C$ with $D \models \phi$ there exists $\delta \in \mathbb{N}$, $D^\delta \models \psi$. Now let $x \in [\![\phi]\!]$ be arbitrary and $D(x) \subseteq C$ the subset

$$D(x) \quad =_{df} \quad \{ V \in C \mid x, V \models_b \phi \},$$

which is time invariant since C is and $x, V \models_b \phi$ implies $x, V^t \models_b \phi$ for all $t \in \mathbb{N}$. $D(x)$ may be empty, but in any case we have that for all $V \in D(x)$, $x, V \models_b \phi$, which by induction hypothesis means $D(x) \models \phi$. Since $D(x) \subseteq C$ we can make use of our assumption to obtain a $\delta \in \mathbb{N}$, $D(x)^\delta \models \psi$. By induction hypothesis again, this implies

$$\exists c \in [\![\psi]\!].\ \forall V \in D(x)^\delta.\ c, V \models_b \psi.$$

Since C is hereditary this is equivalent to the statement that there exists a $c \in [\![\psi]\!]$ such that for all $V \in C$ and $t \in \mathbb{N}$, $x, V^t \models_b \phi$ implies $c, V^{t+\delta} \models_b \psi$. Since we got the existence of c and δ for any x, by the Axiom of Choice, there exists a function $f \in [\![\phi]\!] \to \mathbb{N} \times [\![\psi]\!]$ such that for all $V \in C$ we have

$$\forall x \in [\![\phi]\!].\ \forall t \in \mathbb{N}.\ x, V^t \models_b \phi \ \Rightarrow\ \pi_2\,(f\,x), V^{t+\pi_1\,(f\,x)} \models_b \psi.$$

But this is nothing but the statement that $f, V \models_b \phi \supset \psi$. ∎

We noted that doubly negated formulas are classical statements about the stationary state of a circuit. Such statements, too, have uniform bounds, but they do not contain any information. More precisely, it can be shown that if $c \in [\![\neg\neg\phi]\!]$ and $c, V \models_b \neg\neg\phi$ then for all $d \in [\![\neg\neg\phi]\!]$, $d, V \models_b \neg\neg\phi$. Thus, $\neg\neg\phi$ either has a uniform bound or it does not; and if it does all bounds are uniform bounds.

The following equivalent characterization of validity up to stabilization is useful. We define a syntactic translation of formulas:

$$\begin{array}{rcl}
(a = 1)^\sharp(t, x) & = & \mathit{stable}(V(a), t, 1) \\
(a = 0)^\sharp(t, x) & = & \mathit{stable}(V(a), t, 0) \\
\mathit{false}^\sharp(t, x) & = & \mathit{false} \\
(\phi \wedge \psi)^\sharp(t, x) & = & \phi^\sharp(t, \pi_1\,x) \wedge \psi^\sharp(t, \pi_2\,x) \\
(\phi \vee \psi)^\sharp(t, x) & = & \forall y.\ x = (1, y) \Rightarrow \phi^\sharp(t, y) \wedge \forall y.\ x = (2, y) \Rightarrow \psi^\sharp(t, y) \\
(\phi \supset \psi)^\sharp(t, f) & = & \forall s \geq t.\ \forall x.\ \phi^\sharp(s, x) \Rightarrow \psi^\sharp(s + \pi_1(f\,x), \pi_2(f\,x)).
\end{array}$$

Proposition 5.2 *ϕ is valid for C with bound c iff the formula $\phi^\sharp(0,c)$ is (classically) valid for all valuations $V \in C$.*

Proof: One shows by induction on ϕ that for all $t \in \mathbb{N}$ and $x \in [\![\phi]\!]$, $\phi^\sharp(t,x)$ iff $x, V^t \models_b \phi$. ∎

Thus, a formula ϕ is valid up to stabilization iff there exists a $c \in [\![\phi]\!]$ such that $\phi^\sharp(0,c)$ is a classical tautology.

6 Extracting Stabilization Bounds

When ϕ represents the specification of a combinational circuit, the existence of a timing bound c is reassuring but really we would like to compute and evaluate c in order to obtain quantitative information about the timing behaviour of the circuit. The Completeness Theorem 5.1, which is derived by classical methods, does not *per se* yield a method for finding a bound. Fortunately, often we do not only know that ϕ is valid up to stabilization, but also have a formal proof for it, *i.e.* some extra intensional information about "why" ϕ is true. We will now show that if the proof is a proof in the calculus LJ (*cf.* Sec. 3) we can indeed extract a uniform bound for ϕ from this proof. In general, there may be many different ways of translating proofs into timing bounds. In the sequel we will present one such method that produces useful timing information, and discuss its application to combinational circuits.

We first translate derivations into typed lambda terms with explicit finite sums, finite products, and delay constructs, generated by the syntax

$$t \;\;::=\;\; x \mid \mathit{del}(n) \mid (t,t) \mid \pi_1\, t \mid \pi_2\, t \mid ?(t) \mid \mathit{case}_{x,y}(t,t,t) \mid \iota_1\, t \mid \iota_2\, t \mid t\, t \mid \lambda x.\, t,$$

where n is an arbitrary natural number. The types are

$$\tau \;\;::=\;\; \mathbf{1} \mid \mathbf{0} \mid \tau + \tau \mid \tau \times \tau \mid \tau \to \tau.$$

The formal definition of well-formed terms is as usual and assumed to be understood (*cf.* for example [7]). If $\Delta = x_1{:}\tau_1, \ldots, x_n{:}\tau_n$ is a non-repeating list of variables x_i, each one associated with a type τ_i, we write $\Delta \vdash t : \tau$ to denote that t is a well-formed term of type τ with free variables in Δ. Such a Δ is called a *context.* With two exceptions the type of a term is determined uniquely by the type of its subterms. The type of a variable is generic and thus needs to be given explicitly, as well as the type of the terms $?(t)$. In the last case we write $?^\tau(t)$ to denote this type τ. The nonstandard part of our syntax are the terms $\mathit{del}(n)$, which are the germs for our intensional timing semantics for terms and proofs. They are canonical terms of type $\mathbf{1}$, where the standard λ-calculus only has a single canonical element, say $*$. Intuitively, $\mathit{del}(n)$ represents this standard element $*$ "suffering" from a delay of n time units.
The translation of derivations into terms is controlled by a translation of formulas into types. The type $[\phi]$ associated with a formula ϕ is obtained as follows: $[a = 1] = [a = 0] = \mathbf{1}$, $[\mathit{false}] = \mathbf{0}$, $[\phi \wedge \psi] = [\phi] \times [\psi]$, $[\phi \vee \psi] = [\phi] + [\psi]$, and $[\phi \supset \psi] = [\phi] \to [\psi]$.

Now, given a derivation $D : \phi_n, \ldots, \phi_1 \vdash \psi$, and a context $\Delta = x_1 : [\phi_1], \ldots, x_n : [\phi_n]$ we compute a well-formed term $[D]_\Delta$ of type $[\psi]$ with free variables in Δ as follows:

$$\begin{array}{rcl}
[id]_{x,\Delta} & = & x \\
[get(i) \cdot D]_\Delta & = & [D]_{\Delta^i} \\
[\wedge R \cdot (D_1, D_2)]_\Delta & = & ([D_1]_\Delta, [D_2]_\Delta) \\
[\wedge L \cdot D]_{x,\Delta} & = & [D]_{y_2,y_1,\Delta}\{\pi_2\, x/y_2\}\{\pi_1\, x/y_1\} \\
[\vee L \cdot (D_1, D_2)]_{x,\Delta} & = & case_{y_1,y_2}(x, [D_1]_{y_1,\Delta}, [D_2]_{y_2,\Delta}) \\
[\vee R_1 \cdot D]_\Delta & = & \iota_1\, [D]_\Delta \\
[\vee R_2 \cdot D]_\Delta & = & \iota_2\, [D]_\Delta \\
[falseL]_{x,\Delta} & = & ?(x) \\
[\supset R \cdot D]_\Delta & = & \lambda y.\, [D]_{y,\Delta} \\
[\supset L \cdot (D_1, D_2)]_{x,\Delta} & = & [D_2]_{y,\Delta}\{x\ [D_1]_{x,\Delta}/y\},
\end{array}$$

where Δ^i is the list Δ in which the term at the i-th position has been moved to the front of the list. The variables y, y_1, y_2 are generated by the translation and need to be fresh in each case, *i.e.* must not already occur in Δ and be different from x. Up to the choices of bound variables, the term $[D]_\Delta$ is unique. To enforce this uniqueness we follow the usual custom of identifying terms up to renaming of bound variables.

The next step is to give a denotational semantics for λ-terms so that if D is a proof of ϕ then the denotational semantics of $t = [D]$, which is abbreviated by $[\![t]\!]$, is an element of $[\![\phi]\!]$ and a uniform bound for ϕ. In general, a term t of type τ is mapped into an element $[\![t]\!] \in \mathbb{N} \times [\![\tau]\!]$, where $[\![\tau]\!]$ is defined so that $[\![\mathbf{1}]\!] = [\![\mathbf{0}]\!] = \{*\}$, $[\![\tau_1 \times \tau_2]\!] = [\![\tau_1]\!] \times [\![\tau_2]\!]$, $[\![\tau_1 + \tau_2]\!] = [\![\tau_1]\!] + [\![\tau_2]\!]$, and $[\![\tau_1 \to \tau_2]\!] = [\![\tau_1]\!] \to \mathbb{N} \times [\![\tau_2]\!]$. Note that $[\![[\phi]]\!] = [\![\phi]\!]$.
In order to interpret terms with free variables we need the notion of an *environment*, which is a map ρ that assigns to every variable x of type τ an element $\rho(x) \in \mathbb{N} \times [\![\tau]\!]$. Since for every type τ the set $[\![\tau]\!]$ is nonempty we are entitled to assume for every τ a predefined choice of an element $\kappa_\tau \in [\![\tau]\!]$. The semantics is given by the following inductive definition:

- $[\![x]\!]_\rho = \rho(x)$
- $[\![del(\delta)]\!]_\rho = (\delta, *)$
- If $[\![t]\!]_\rho = (\delta, v)$ then $[\![?^\tau(t)]\!]_\rho = (\delta, \kappa_\tau)$
- If $[\![t_1]\!]_\rho = (\delta_1, v_1)$ and $[\![t_2]\!]_\rho = (\delta_2, v_2)$ then $[\![(t_1, t_2)]\!]_\rho = (max(\delta_1, \delta_2), (v_1, v_2))$
- If $[\![t]\!]_\rho = (\delta, (v_1, v_2))$ then $[\![\pi_i\, t]\!]_\rho = (\delta, v_i)$, for $i = 1, 2$
- If $[\![t]\!]_\rho = (\delta, v)$ then $[\![\iota_i\, t]\!]_\rho = (\delta, (i, v))$, for $i = 1, 2$
- If $[\![t]\!]_\rho = (\delta_i, (i, v_i))$ and $[\![t_i]\!]_{\rho[x_i \leftarrow (0, v_i)]} = (\delta, v)$, then $[\![case_{x_1,x_2}(t, t_1, t_2)]\!]_\rho = (\delta_i + \delta, v)$, for $i = 1, 2$
- $[\![\lambda x.\, t]\!]_\rho = (0, f)$ where f is the function that maps every element $v \in [\![\tau_1]\!]$, τ_1 being the type of x, to the value $[\![t]\!]_{\rho[x \leftarrow (0,v)]}$

- If $t_1 : \tau_2 \to \tau_1$ and $t_2 : \tau_2$ such that $[\![t_1]\!]_\rho = (\delta_1, f_1)$ with $f_1 \in [\![\tau_2]\!] \to \mathbb{N} \times [\![\tau_1]\!]$, and $[\![t_2]\!]_\rho = (\delta_2, v_2)$, and $f_1(v_2) = (\delta, v)$, then $[\![t_1\, t_2]\!]_\rho = (\delta + max(\delta_1, \delta_2), v)$.

Proposition 6.1 (Correctness) *Let t be a well-formed term of type τ. Then for every environment ρ the above inductive rules define a uniquely determined element $[\![t]\!]_\rho \in \mathbb{N} \times [\![\tau]\!]$.*

Proof: Simple, by induction on the structure of t. ∎

If $\pi_1\, [\![t]\!]_\rho = 0$ then t is called *stable* in environment ρ. For a stable term $t : \tau$ we may confuse $[\![t]\!]_\rho \in \mathbb{N} \times [\![\tau]\!]$ and $\pi_2\, [\![t]\!]_\rho \in [\![\tau]\!]$.

Lemma 6.2 (Substitution Lemma) *Let $t_1 : \tau_1$ and $t_2 : \tau_2$ well-formed terms and $x : \tau_1$ a free variable of t_2. Then $[\![t_2\{t_1/x\}]\!]_\rho = [\![t_2]\!]_{\rho[x \leftarrow [\![t_1]\!]_\rho]}$.*

Proof: By induction on t_2. It is important to be aware that the substitution $t_2\{t_1/x\}$ possibly needs to rename bound variables in order to avoid name capture. ∎

Lemma 6.3 (Soundness Lemma) *Let $D : \Gamma \vdash \psi$ be a derivation with $\Gamma = \phi_n, \ldots, \phi_1$, and $\Delta = x_1 : [\phi_1], \ldots, x_n : [\phi_n]$ a context of variables for Γ. Further let $t_D = [D]_\Delta$ be the extracted λ-term for D. Then for all environments ρ, all $t \in \mathbb{N}$, and all valuations V,*

$$\forall i \leq n.\ \pi_2\, \rho(x_i),\ V^{t + \pi_1\, \rho(x_i)} \models_b \phi_i \quad \Rightarrow \quad \pi_2\, [\![t_D]\!]_\rho,\ V^{t + \pi_1\, [\![t_D]\!]_\rho} \models_b \psi.$$

Proof: Let us first introduce some useful notation. For $c \in \mathbb{N} \times [\![\phi]\!]$ and $V \in \mathbb{S} \to \mathbb{N} \to \mathbb{B}$ we define $c, V \models_b^* \phi$ to stand for the condition $\pi_2\, c, V^{\pi_1\, c} \models_b \phi$. Now, let $D : \Gamma \vdash \psi$ be a derivation with $\Gamma = \phi_n, \ldots, \phi_1$, and $\Delta = x_1 : [\phi_1], \ldots, x_n : [\phi_n]$ a context of variables for Γ. Let us take $[\![D]\!]_\rho^\Delta$ to abbreviate $[\![[D]_\Delta]\!]_\rho$, and $\rho(\Delta), V \models_b^* \Gamma$ to abbreviate $\forall i \leq n.\ \rho(x_i),\ V \models_b^* \phi_i$. The proof that for all ρ, t, V,

$$\rho(\Delta), V^t \models_b^* \Gamma \;\Rightarrow\; [\![D]\!]_\rho^\Delta, V^t \models_b^* \psi$$

is by induction of the structure of D. We will only treat the cases where D ends in an application of $\supset\! L$, $\supset\! R$, or $\vee L$. All other cases, *viz.* rules in $\{id, get(i), \wedge R, \wedge L, \vee R_1, \vee R_2, falseL\}$ are omitted as they are rather simple.

- Suppose D ends in an application of the rule $\supset\! L$, *i.e.* $D = \supset\! L \cdot (D_1, D_2)$ with the associated proof tree

$$\frac{D_1 : \Gamma, \phi \supset \psi \vdash \phi \qquad D_2 : \Gamma, \psi \vdash \theta}{D : \Gamma, \phi \supset \psi \vdash \theta} \supset\! L$$

Let Δ be an arbitrary, but fixed, context for Γ and $x : [\phi \supset \psi]$. The goal is to show

$$\forall \rho, t, V.\quad \rho(\Delta), V^t \models_b^* \Gamma \quad \& \quad \rho(x), V^t \models_b^* \phi \supset \psi \;\;\Rightarrow\;\; [\![D]\!]_\rho^{x,\Delta}, V^t \models_b^* \theta.$$

To this end let ρ, t, and V be fixed, and assume

$$\textit{(i)}\quad \rho(\Delta), V^t \models_b^* \Gamma \qquad\qquad \textit{(ii)}\quad \rho(x), V^t \models_b^* \phi \supset \psi.$$

We wish to show that $[\![D]\!]_\rho^{x,\Delta}, V^t \models_b^* \theta$, where $[\![D]\!]_\rho^{x,\Delta}$ is determined as follows: By definition of the translation $[D]_{x,\Delta} = [\supset L \cdot (D_1, D_2)]_{x,\Delta} = [D_2]_{y,\Delta}\{x\,[D_1]_{x,\Delta}/y\}$, where y is some fresh variable of type $[\psi]$ different from x and not occurring in Δ. Now, by the Substitution Lemma 6.2, we get $[\![D]\!]_\rho^{x,\Delta} = [\![D_2]\!]_{\rho[y\leftarrow c]}^{y,\Delta}$ where $c = [\![x\,[D_1]_{x,\Delta}]\!]_\rho$. Using our semantic definition we find that $c = (\delta, v)$ such that

$$\begin{array}{rcl} \delta & = & \pi_1\,((\pi_2\,\rho(x))(\pi_2\,[\![D_1]\!]_\rho^{x,\Delta})) \;+\; max(\pi_1\,\rho(x), \pi_1[\![D_1]\!]_\rho^{x,\Delta}) \\ v & = & \pi_2\,((\pi_2\,\rho(x))(\pi_2\,[\![D_1]\!]_\rho^{x,\Delta})). \end{array}$$

Thus our goal is to verify

$$[\![D_2]\!]_{\rho[y\leftarrow c]}^{y,\Delta}, V^t \models_b^* \theta. \tag{7}$$

The proof of (7) may use the induction hypothesis for both $D_1 : \Gamma, \phi \supset \psi \vdash \phi$ with context x, Δ and $D_2 : \Gamma, \psi \vdash \theta$ with context y, Δ, *viz.*

$$\begin{array}{llcll} \forall\rho, t, V. & \rho(\Delta), V^t \models_b^* \Gamma & \& & \rho(x), V^t \models_b^* \phi \supset \psi & \Rightarrow\; [\![D_1]\!]_\rho^{x,\Delta}, V^t \models_b^* \phi \\ \forall\rho, t, V. & \rho(\Delta), V^t \models_b^* \Gamma & \& & \rho(y), V^t \models_b^* \psi & \Rightarrow\; [\![D_2]\!]_\rho^{y,\Delta}, V^t \models_b^* \theta. \end{array}$$

By choosing the ρ in the second induction hypothesis to be $\rho[y \leftarrow c]$ we can reduce our goal (7) to

$$\rho[y \leftarrow c](\Delta), V^t \models_b^* \Gamma \quad \& \quad c, V^t \models_b^* \psi.$$

The first of these two conditions is immediate from *(i)*, observing that the values of both environments $\rho[y \leftarrow c]$ and ρ on the context Δ are the same. Thus we are left with the problem of showing

$$c, V^t \models_b^* \psi. \tag{8}$$

This is done as follows. We first use assumptions *(i)* and *(ii)* and the first induction hypothesis to infer $[\![D_1]\!]_\rho^{x,\Delta}, V^t \models_b^* \phi$, which more explicitly is

$$\textit{(iii)}\quad \pi_2\,[\![D_1]\!]_\rho^{x,\Delta}, V^{t+\pi_1\,[\![D_1]\!]_\rho^{x,\Delta}} \models_b \phi.$$

Now, by definition of $\models_b^*$ *(ii)* implies that for all $(s, d) \in \mathbb{N} \times [\![\phi]\!]$,

$$d, V^{t+\pi_1\,\rho(x)+s} \models_b \phi \;\;\Rightarrow\;\; (\pi_2\,\rho(x))\,d, V^{t+\pi_1\,\rho(x)+s} \models_b^* \psi. \tag{9}$$

In particular we may choose $d = \pi_2\,[\![D_1]\!]_\rho^{x,\Delta}$ and $s = max(\pi_1\,[\![D_1]\!]_\rho^{x,\Delta} - \pi_1\,\rho(x), 0)$. Then $s \in \mathbb{N}$ and $s + \pi_1\,\rho(x) \geq \pi_1\,[\![D_1]\!]_\rho^{x,\Delta}$, *i.e.* by *(iii)*, $\pi_2\,[\![D_1]\!]_\rho^{x,\Delta}, V^{t+\pi_1\,\rho(x)+s} \models_b \phi$. Now we can apply (9) to obtain $(\pi_2\,\rho(x))\,(\pi_2\,[\![D_1]\!]_\rho^{x,\Delta}), V^{t+\pi_1\,\rho(x)+s} \models_b^* \psi$. But since $\pi_1\,\rho(x) + s = max(\pi_1\,[\![D_1]\!]_\rho^{x,\Delta}, \pi_1\,\rho(x))$, this is nothing but (8) as desired.

• Suppose D ends in an application of the rule $\supset R$, *i.e.* $D = \supset R \cdot D_1$ with the associated proof tree

$$\frac{D_1 : \Gamma, \phi \vdash \psi}{D : \Gamma \vdash \phi \supset \psi} \supset R$$

Let Δ be an arbitrary but fixed context for Γ. Let ρ, t, V be given such that *(i)* $\rho(\Delta), V^t \models_b^* \Gamma$. We must show that

$$[\![D]\!]_\rho^\Delta, V^t \models_b^* \phi \supset \psi. \tag{10}$$

We compute $[D]_\Delta = [\supset R \cdot D_1]_\Delta = \lambda x.\ [D_1]_{x,\Delta}$, where x is a fresh variable of type $[\phi]$. Thus, by the semantics clauses, $[\![D]\!]_\rho^\Delta = [\![\lambda x.\ [D_1]_{x,\Delta}]\!]_\rho = (0, f)$, where f is the function that maps every $c \in [\![\phi]\!]$ to the element $f(c) = [\![D_1]\!]_{\rho[x \leftarrow (0,c)]}^{x,\Delta}$. With this information we find that (10) is equivalent to the statement

$$\forall d \in \mathbb{N} \times [\![\phi]\!].\ d, V^t \models_b^* \phi \ \Rightarrow\ [\![D_1]\!]_{\rho[x \leftarrow (0, \pi_2\, d)]}^{x,\Delta}, V^{t + \pi_1\, d} \models_b^* \psi.$$

So, let $d \in \mathbb{N} \times [\![\phi]\!]$ with *(ii)* $d, V^t \models_b^* \phi$ be given. We wish to show

$$[\![D_1]\!]_{\rho[x \leftarrow (0, \pi_2\, d)]}^{x,\Delta}, V^{t + \pi_1\, d} \models_b^* \psi. \tag{11}$$

We employ the induction hypothesis for $D_1 : \Gamma, \phi \vdash \psi$ in context x, Δ,

$$\forall \rho, t, V.\quad \rho(\Delta), V^t \models_b^* \Gamma \quad \& \quad \rho(x), V^t \models_b^* \phi \ \Rightarrow\ [\![D_1]\!]_\rho^{x,\Delta}, V^t \models_b^* \psi$$

with the particular environment $\rho[x \leftarrow (0, \pi_2\, d)]$ and time $t + \pi_1\, d$ to reduce our goal (11) to

$$\rho[x \leftarrow (0, \pi_2\, d)](\Delta), V^{t+\pi_1\, d} \models_b^* \Gamma \quad \& \quad (0, \pi_2\, d), V^{t+\pi_1\, d} \models_b^* \phi.$$

The second condition is just our assumption *(ii)*, while the first follows from *(i)* by observing that ρ and $\rho[x \leftarrow (0, \pi_2\, d)]$ obtain the same values in context Δ, and that $\rho(\Delta), V^t \models_b^* \Gamma$ implies $\rho(\Delta), V^{t+\pi_1\, d} \models_b^* \Gamma$. Thus, we have verified (11), which completes the proof.

• Suppose ends in an application of rule $\vee L$, *i.e.* $D = \vee L \cdot (D_1, D_2)$ with the associated proof tree

$$\frac{D_1 : \Gamma, \phi \vdash \theta \qquad D_2 : \Gamma, \psi \vdash \theta}{D : \Gamma, \phi \vee \psi \vdash \theta} \vee L$$

Let Δ be a context for Γ and x a variable of type $[\phi \vee \psi] = [\phi] + [\psi]$ not occurring in Δ. Let environment ρ, $t \in \mathbb{N}$ and valuation V be fixed such that *(i)* $\rho(\Delta), V^t \models_b^* \Gamma$ and *(ii)* $\rho(x), V^t \models_b^* \phi \vee \psi$. We have to check that

$$[\![D]\!]_\rho^{x,\Delta}, V^t \models_b^* \theta. \tag{12}$$

We prove this by case analysis on $\rho(x) \in \mathbb{N} \times ([\![\phi]\!] + [\![\psi]\!])$. The first case is that $\rho(x) = (\delta_1, (1, v_1))$ for $\delta_1 \in \mathbb{N}$ and $v_1 \in [\![\phi]\!]$. We compute

$$[D]_{x,\Delta} = [\vee L \cdot (D_1, D_2)]_{x,\Delta} = case_{y_1,y_2}(x, [D_1]_{y_1,\Delta}, [D_2]_{y_2,\Delta})$$

where y_1, y_2 are fresh variables different from x and not ocurring in Δ. Then, with $\rho(x) = (\delta_1, (1, v_1))$, we get $[\![D]\!]_\rho^{x,\Delta} = (\delta_1 + \pi_1 [\![D_1]\!]^{y_1,\Delta}_{\rho[y_1 \leftarrow (0,v_1)]}, \pi_2 [\![D_1]\!]^{y_1,\Delta}_{\rho[y_1 \leftarrow (0,v_1)]})$. This means, our goal (12) becomes

$$[\![D_1]\!]^{y_1,\Delta}_{\rho[y_1 \leftarrow (0,v_1)]}, V^{t+\delta_1} \models_b^* \theta. \tag{13}$$

Using the induction hypothesis for $D_1 : \Gamma, \phi \vdash \theta$ this can be reduced to

$$\rho[y_1 \leftarrow (0, v_1)](\Delta), V^{t+\delta_1} \models_b^* \Gamma \quad \& \quad (0, v_1), V^{t+\delta_1} \models_b^* \phi.$$

The second condition follows from *(ii)* and the first from *(i)*. This finishes the case that $\rho(x) = (\delta_1, (1, v_1))$. The other case, $\rho(x) = (\delta_2, (2, v_2))$ is symmetrical, using the induction hypothesis for D_2. ∎

Theorem 6.4 *Let D be a proof of ϕ and $t_D = [D]$ the corresponding closed λ-term for D. Then t_D is stable and $[\![t_D]\!]$ is a uniform bound for ϕ.*

Proof: The first part of the theorem claims that the extracted term t_D for a proof D is stable, *i.e.* $\pi_1 [\![t_D]\!] = 0$. This is easy to see since the extracted term t_D does not contain any subterm of form $del(\delta)$ which are the only terms that could introduce a nonzero delay value into the term. The rest of the theorem, then, follows from the previous Lem. 6.3 as a special case. ∎

With Thm. 6.4 the goal of this section, *viz.* to compute uniform stabilization bounds for theorems of LJ, is achieved. The remaining task now is to demonstrate that the bounds obtained by this method produce the desired results in concrete circuit verifications. This will be done in the next section.

Before we turn to applications it appears appropriate to spend some words on semantical issues. The obvious question one might ask is what kind of semantics $[\![\_]\!]$ induces on our terms and *a fortiori* on our proofs in LJ, and whether indeed this justifies calling it a lambda calculus. In fact, it turns out that we get a computational lambda calculus in the sense of Moggi [14], where the existence predicate $t \downarrow$ is interpreted as the statement that t is stable.

Proposition 6.5 *Let t be stable. Then,*

$$\begin{array}{rcccl}
[\![(\lambda x.\, s)\, t]\!]_\rho & = & [\![s\{t/x\}]\!]_\rho & & \\
[\![\pi_2(t, s)]\!]_\rho & = & [\![s]\!]_\rho & = & [\![\pi_1(s, t)]\!]_\rho \\
[\![case_{x,y}(\iota_1\, t, s_1, s_2)]\!]_\rho & = & [\![s_1\{t/x\}]\!]_\rho & = & [\![case_{y,x}(\iota_2\, t, s_2, s_1)]\!]_\rho.
\end{array}$$

Proof: By simple check of definition and application of Substitution Lemma 6.2. ∎

Prop. 6.5 are the standard β-reductions of λ-calculus. The side condition that t be stable is necessary in the sense that all equations have (simple) counter examples where t is not stable. We also have the usual η-equations:

Proposition 6.6 *Let t be stable. Then,*

$$\begin{aligned}
[\![\lambda x.\,(t\,x)]\!]_\rho &= [\![t]\!]_\rho \\
[\![(\pi_1\,t, \pi_2\,t)]\!]_\rho &= [\![t]\!]_\rho \\
[\![case_{x,y}(t, \iota_1\,x, \iota_2\,y)]\!]_\rho &= [\![t]\!]_\rho \\
[\![?^{\mathbf{0}}(t)]\!]_\rho &= [\![t]\!]_\rho.
\end{aligned}$$

Proof: By simple check of definition. ∎

As far as the η-equations are concerned the side condition of t being stable is only needed for lambda abstraction.

Our semantics can be seen as an interpretation in a particular *computational λ-model* [14]: *viz.* in the category of sets with a strong monad $T : A \mapsto \mathbb{N} \times A$ that has unit $\eta_A : A \to \mathbb{N} \times A$ with $\eta(a) = (0, a)$, multiplication $\mu_A : \mathbb{N} \times \mathbb{N} \times A \to \mathbb{N} \times A$ with $\mu(n_1, n_2, a) = (n_1 + n_2, a)$, and tensorial strength $\sigma_{A,B} : (\mathbb{N} \times A \times \mathbb{N} \times B) \to \mathbb{N} \times A \times B)$ defined as $\mu(n_1, a, n_2, b) = (max(n_1, n_2), a, b)$. Within our λ-calculus this special monad is reflected in the structure of the terms of type $\mathbf{1}$: They can be viewed as elements of $\mathbb{N}$ endowed with operations $+$ and max. Given terms $s, t : \mathbf{1}$ we can define $s + t =_{df} (\lambda x.\ s)\,t$ and $max(s, t) = \pi_1\,(s, t)$. These syntactic operations satisfy the standard properties of addition and maximum. In particular, we have $[\![del(n) + del(m)]\!] = [\![del(n + m)]\!]$ and $[\![max(del(n), del(m))]\!] = [\![del(max(n, m))]\!]$.

In the context of functional programming languages the intensional view is a natural idea. For instance, Dough Gurr [9] considers non-extensional semantics for a lambda calculus in which explicit complexity information is added. Our delay time, at least *grosso modo*, may be seen as a special case. However, Gurr's notion of complexity only involves the operations of a monoid whereas for our purposes a richer delay algebra, *viz.* $(\mathbb{N}, 0, +, max)$, is needed. Both operations $+$ and max are essential to derive sensible timing information, as we will see in the next section. Thus, our framework applies a more refined view of (time) complexity than Gurr's.

7 Application to Combinational Circuits

We have implemented an experimental prototype system for the timing analysis of combinational circuits based on the ideas presented in the previous sections. Here we wish to illustrate some of the basic ideas on which the potential application of our experimental system rests. In particular, our aim is to demonstrate that

1. in the presence of propagation delays intuitionistic functional verification may be more adequate than classical two-valued reasoning, without being cluttered up by timing details. This contrasts with the other options available, *e.g.* using classical temporal logic.

2. the stabilization bounds obtained from a proof yield data-dependent timing information of the circuit that has been verified, *i.e.* the computed propagation delays are specific to a particular function and particular input stimulus of the circuit. This contrasts with computing the worst-case (= topological) delay, *i.e.* the longest path through the circuit.

To keep matters short we will not enlarge on the application in general, but focus on the simple combinational circuit shown in Fig. 2 as a suggestive example, leaving the generalization to the imagination of the reader. We wish to stress that all the analysis steps presented in this section can be done automatically by our implementation.

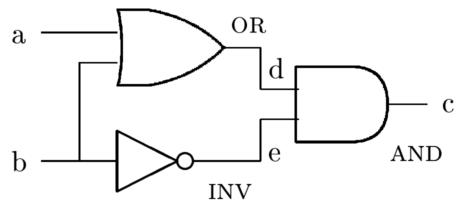

Figure 2: A Simple Combinational Circuit CIRC.

Specifications The circuit's behaviour, as well as the behaviour of its components, is captured by formulas of LJ. For instance, the OR gate in Fig. 2 which has inputs a and b and output d may be specified by

$$\mathrm{OR} \;=\; \mathrm{OR}\uparrow \wedge \mathrm{OR}\downarrow \;=\; ((a=1 \vee b=1) \supset d=1) \;\wedge\; ((a=0 \wedge b=0) \supset d=0).$$

In a sense, this specification is nothing but the input-output function table of the OR: The first conjunct OR $\uparrow$ captures the conditions for a rising output transition: "If *one* of the inputs a and b is stable 1, then this gives rise to output d being stable 1 in bounded time." The second conjunct OR $\downarrow$ sums up our assumption about the falling output transition: "If *both* a and b are stable 0, then in bounded time d is stable 0." Let us analyse this in more detail for the rising output transition of OR. By Thm. 5.1 a circuit C satisfies OR $\uparrow$ iff there exists a stabilization bound $c \in [\![\mathrm{OR}\uparrow]\!] = \{*\} + \{*\} \to \mathbb{N} \times \{*\}$ such that OR $\uparrow$ is valid for C with bound c. This, by Prop. 5.2 means that for all $V \in C$ the following formula is true:

$$\begin{aligned}
&(\mathrm{OR}\uparrow)^\sharp(0,c)\\
&\quad= \;((a=1 \vee b=1) \supset d=1)^\sharp(0,c)\\
&\quad= \;\forall t.\,\forall x.\,(a=1 \vee b=1)^\sharp(t,x) \Rightarrow (d=1)^\sharp(t+\pi_1(c\,x),\pi_2(c\,x))\\
&\quad= \;\forall t.\,\forall x.\\
&\qquad\qquad (\forall y.\, x=(1,y) \Rightarrow (a=1)^\sharp(t,y) \;\wedge\; \forall y.\, x=(2,y) \Rightarrow (b=1)^\sharp(t,y))\\
&\qquad\qquad\quad \Rightarrow (d=1)^\sharp(t+\pi_1(c\,x),\pi_2(c\,x))\\
&\quad\equiv \;\forall t.\,((a=1)^\sharp(t,*) \Rightarrow (d=1)^\sharp(t+\pi_1(c\,(1,*)),\pi_2(c\,(1,*)))) \;\wedge
\end{aligned}$$

$$((b=1)^\sharp(t,*) \Rightarrow (d=1)^\sharp(t+\pi_1(c\,(2,*)),\pi_2(c\,(2,*))))$$
$$= \forall t.\ (\mathit{stable}(V(a),t,1) \Rightarrow \mathit{stable}(V(d),t+\pi_1(c\,(1,*)),1)) \ \wedge$$
$$(\mathit{stable}(V(b),t,1) \Rightarrow \mathit{stable}(V(d),t+\pi_1(c\,(2,*)),1)).$$

The forth equivalence uses the fact that $\forall x \in \{*\}+\{*\}.\ \phi(x)$ is semantically the same as $\phi((1,*))\ \&\ \phi((2,*))$. Note that $\{*\}+\{*\} \to \mathbb{N}\times\{*\} \cong \mathbb{N}\times\mathbb{N}$, so that c is uniquely determined by two numbers $\delta^1_{\text{OR}\uparrow} = \pi_1(c\,(1,*))$ and $\delta^2_{\text{OR}\uparrow} = \pi_1(c\,(2,*))$. Our derivation shows that if c is the stabilization bound of the OR gate then $\delta^1_{\text{OR}\uparrow}$ and $\delta^2_{\text{OR}\uparrow}$ are the propagation delays for a rising output transition triggered by the first and second input, respectively. For the falling output transition OR $\downarrow$, in contrast, a stabilization bound $c \in [\![\text{OR}\downarrow]\!] = \{*\}\times\{*\} \to \mathbb{N}\times\{*\} \cong \mathbb{N}$ is a single number $\delta_{\text{OR}\downarrow}$, *viz.* $\delta_{\text{OR}\downarrow} = \pi_2(c(*,*))$. To sum up we find that $C \models \text{OR}$ is equivalent to the existence of a triple of natural numbers $(\delta^1_{\text{OR}\uparrow}, \delta^2_{\text{OR}\uparrow}, \delta_{\text{OR}\downarrow})$ such that

$$\begin{aligned}
&\text{OR}^\sharp(0,c)\\
&\quad= \forall t.\ (\mathit{stable}(V(a),t,1) \Rightarrow \mathit{stable}(V(d),t+\delta^1_{\text{OR}\uparrow},1)) \ \wedge\\
&\qquad\quad (\mathit{stable}(V(b),t,1) \Rightarrow \mathit{stable}(V(d),t+\delta^2_{\text{OR}\uparrow},1)) \ \wedge\\
&\qquad \forall t.\ \mathit{stable}(V(a),t,0) \wedge \mathit{stable}(V(b),t,0) \Rightarrow \mathit{stable}(V(d),t+\delta_{\text{OR}\downarrow},0)
\end{aligned}$$

is true for all $V \in C$. This means that a stabilization bound for OR may be viewed as a data-dependent delay table distinguishing between rising and falling output transitions and between different inputs. In practice these differences do occur, and for instance in CMOS technology where one may face a relative variation of more than 50% [8]. Accounting for these differences in the timing model is crucial in wave pipelining applications or in the optimization of synchronous designs [1]. Even if the primitive gates are modeled with single fixed delays, the propagation delay of a composite circuit, in general, will be data-dependent, since not all subcomponents are used for every function of the circuit.

It is useful to have a syntactic representation for the stabilization bounds, alias delay tables. In fact one can show that

$$T_{\text{OR}} : [\text{OR}] \ =_{df}\ (\lambda y.\ \mathit{case}_{x_1,x_2}(y, \mathit{del}(\delta^1_{\text{OR}\uparrow}), \mathit{del}(\delta^2_{\text{OR}\uparrow}))\ ,\ \lambda y.\ \mathit{del}(\delta_{\text{OR}\downarrow}))$$

is the generic λ-term for $[\![\text{OR}]\!]$, *i.e.* for every element $c \in [\![\text{OR}]\!]$ there exist $\delta^1_{\text{OR}\uparrow}, \delta^2_{\text{OR}\uparrow}, \delta_{\text{OR}\downarrow}$ such that $[\![T_{\text{OR}}]\!] = c$.

To complete our specification of the gates, here are the formulas for AND and INV

$$\begin{array}{lclcl}
\text{AND} & = & \text{AND}\uparrow \wedge \text{AND}\downarrow & = & (d=1 \wedge e=1 \supset c=1)\ \wedge\ (d=0 \vee e=0 \supset c=0)\\
\text{INV} & = & \text{INV}\uparrow \wedge \text{INV}\downarrow & = & (b=0 \supset e=1)\ \wedge\ (b=1 \supset e=0),
\end{array}$$

which again are nothing but the formula-ization of their respective input-output function tables. Typical instances of associated delay tables are

$$\begin{array}{lcl}
T_{\text{AND}} & = & (\lambda y.\ \delta_{\text{AND}\uparrow}\ ,\ \lambda y.\ \mathit{case}_{x_1,x_2}(y, \delta^1_{\text{AND}\downarrow}, \delta^2_{\text{AND}\downarrow}))\\
T_{\text{INV}} & = & (\lambda y.\ \delta_{\text{INV}\uparrow}\ ,\ \lambda y.\ \delta_{\text{INV}\downarrow}).
\end{array}$$

Let us consider the behaviour of the composite circuit CIRC of Fig. 2. If we used classical 2-valued logic, then, regarding a falling output transition of the circuit, we would be entitled to conclude that $a = 0$ implies $c = 0$. For if $a = 0$ then no matter which value input b assumes, one of the two inputs of the AND is 0. However, in reality this reasoning is deceptive. In fact, assuming $a = 0$, any $1 \to 0$ transition on b may produce a spurious $0 \to 1 \to 0$ glitch at output c, which contradicts $c = 0$. The glitch occurs when the propagation of the $1 \to 0$ edge through the INV is faster than through the OR gate (which is not unreasonable). The flaw in the classical reasoning, of course, is the hidden assumption that input b is either stable 0 or stable 1. Hence, to mend the argument we must enforce explicitly that b is stable. But this is not expressible in a two-valued logic. In our intuitionistic logic this *can* be expressed, *viz.* by the formula $b = 1 \vee b = 0$. The output transitions of the circuit, then, are safely specified by

$$\begin{aligned} \text{CIRC} &= \text{CIRC}\uparrow \ \wedge \text{CIRC}\downarrow \\ &= (a = 1 \wedge b = 0 \supset c = 1) \ \wedge \ (a = 0 \wedge (b = 0 \vee b = 1) \supset c = 0). \end{aligned} \tag{14}$$

Formal Verification The verification goal is the construction of a proof that the composition of the AND, OR, and INV gates results in a circuit satisfying CIRC, *i.e.* we seek a formal derivation of the sequent

$$\vdash (\text{OR} \wedge \text{INV} \wedge \text{AND}) \supset \text{CIRC}. \tag{15}$$

A search for this derivation would apply the rules of LJ given in Fig. 1 in the usual backward manner by unification, and eventually end up with a complete derivation, such as the one seen in Fig. 3. Here, for the sake of compactness some obvious abbreviations have been adopted. For instance, $\text{O}, \text{I}, \text{A}$ stand for the formulas $\text{OR}, \text{INV}, \text{AND}$, respectively. The sub-derivations

$$\begin{aligned} D_1 &: \text{OR}, \text{INV}, \text{AND}, \vdash \text{CIRC}\uparrow \\ D_2 &: \text{OR}, \text{INV}, \text{AND}, \vdash \text{CIRC}\downarrow \end{aligned}$$

indicated in Fig. 3 establish the two main parts of the circuit's input-output function table. They are composed in an appropriate way to form a derivation D of sequent (15.

If we were concerned only with the functional behaviour of the circuit we could stop at this point. We have certainly invested more work compared to a classical two-valued analysis, but in return we get a verified statement about the functionality of the circuit that is safe even in the presence of (possibly unbounded) propagation delays and oscillations.

Translation into Lambda Terms The main advantage of the proposed constructive framework is that we can derive timing information, *viz.* stabilization bounds, from the derivations, which is not possible in a classical two-valued world. Since all relevant information is already contained in D_1 and D_2 we will focus on these sub-derivations in the sequel.

Intuitively, the derivations D_1 and D_2 not only witness the mere fact that the composition of the components satisfies the functionality of CIRC, but also record the information just

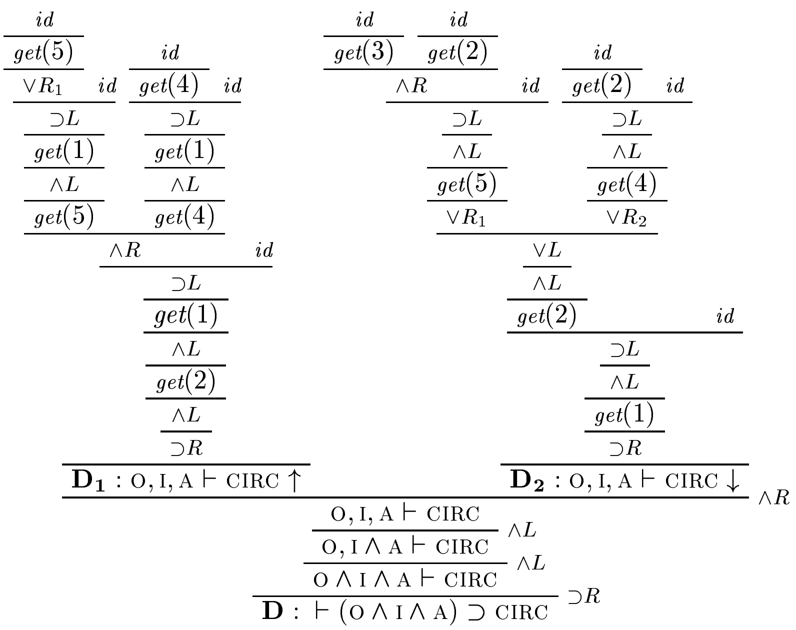

Figure 3: A Possible Proof.

how the goal is met. More precisely, for each part of the functional specification CIRC it is recorded how this part is achieved, what components are used, and what aspect of a component's functionality is involved. Our translation of the derivation into a λ-term, then, extracts this extra information: It represents a timing function which translates the delay tables of OR, INV, and AND to produce a delay table of CIRC.

The translation follows Sec. 6. Since the D_i are open derivations, *i.e.* with premisses on the left-hand side of the sequent turn stile, the λ-terms obtained are open terms, with free variables being the parameters that represent stabilization bounds of the sequent's premisses. Let x_O : [OR], x_I : [INV], and x_A : [AND] be variables denoting stabilization bounds for OR, INV, and AND, respectively. Then the translation as defined in Sec. 6 applied to D_1 yields:

$$
\begin{aligned}
&[D_1]_{x_A\,,\,x_I\,,\,x_O} \\
&\quad = \; [\supset R \cdot \wedge L \cdot get(2) \cdot \wedge L \cdot get(1) \cdot \supset L \cdot (\wedge R \cdots\ ,\ id)]_{x_A\,,\,x_I\,,\,x_O} \\
&\quad = \; \lambda y.\, [\wedge L \cdot get(2) \cdot \wedge L \cdot get(1) \cdot \supset L \cdot (\wedge R \cdots\ ,\ id)]_{y\,,\,x_A\,,\,x_I\,,\,x_O} \\
&\quad = \; \lambda y.\, [get(2) \cdot \wedge L \cdot get(1) \cdot \supset L \cdot (\wedge R \cdots\ ,\ id)]_{y_2\,,\,y_1\,,\,x_A\,,\,x_I\,,\,x_O} \\
&\qquad\qquad \{\pi_2\, y/y_2\}\{\pi_1\, y/y_1\} \\
&\quad = \; \lambda y.\, [\wedge L \cdot get(1) \cdot \supset L \cdot (\wedge R \cdots\ ,\ id)]_{x_A\,,\,y_2\,,\,y_1\,,\,x_I\,,\,x_O} \\
&\qquad\qquad \{\pi_2\, y/y_2\}\{\pi_1\, y/y_1\} \\
&\quad = \; \lambda y.\, [get(1) \cdot \supset L \cdot (\wedge R \cdots\ ,\ id)]_{x_2\,,\,x_1\,,\,y_2\,,\,y_1\,,\,x_I\,,\,x_O}
\end{aligned}
$$

$$
\begin{array}{rl}
& \{\pi_2\, x_A/x_2\}\{\pi_1\, x_A/x_1\}\{\pi_2\, y/y_2\}\{\pi_1\, y/y_1\} \\
= & \lambda y.\, [\supset\! L \cdot (\wedge R \cdot \cdots\ ,\ id)]_{x_1\,,\,x_2\,,\,y_2\,,\,y_1\,,\,x_I\,,\,x_O} \\
& \{\pi_2\, x_A/x_2\}\{\pi_1\, x_A/x_1\}\{\pi_2\, y/y_2\}\{\pi_1\, y/y_1\} \\
= & \lambda y.\, [id]_{x\,,\,x_2\,,\,y_2\,,\,y_1\,,\,x_I\,,\,x_O} \\
& \{x_1\, [\wedge R \cdot \cdots]_{x_1\,,\,x_2\,,\,y_2\,,\,y_1\,,\,x_I\,,\,x_O}/x\}\{\pi_2\, x_A/x_2\}\{\pi_1\, x_A/x_1\}\{\pi_2\, y/y_2\}\{\pi_1\, y/y_1\} \\
= & \lambda y.\, x \\
& \{x_1\, [\wedge R \cdot \cdots]_{x_1\,,\,x_2\,,\,y_2\,,\,y_1\,,\,x_I\,,\,x_O}/x\}\{\pi_2\, x_A/x_2\}\{\pi_1\, x_A/x_1\}\{\pi_2\, y/y_2\}\{\pi_1\, y/y_1\} \\
= & \lambda y.\, (\pi_1\, x_A)\, [\wedge R \cdot \cdots]_{x_1\,,\,x_2\,,\,y_2\,,\,y_1\,,\,x_I\,,\,x_O} \\
& \{\pi_2\, x_A/x_2\}\{\pi_1\, x_A/x_1\}\{\pi_2\, y/y_2\}\{\pi_1\, y/y_1\} \\
\vdots & \quad \vdots \\
= & \lambda y.\, (\pi_1\, x_A)\, ((\pi_1\, x_O)\, (\iota_1\, (\pi_1\, y))\ ,\ (\pi_1\, x_I)\, (\pi_2\, y)).
\end{array}
$$

One verifies that this is a well-formed term of type $\mathbf{1}\times\mathbf{1} \to \mathbf{1}$. In a similar way we translate the other derivation and obtain:

$$
[D_2]_{x_A\,,\,x_I\,,\,x_O} \;=\; \lambda y.\, (\pi_2\, x_A)\, case_{x_1,x_2}(\pi_2\, y, \iota_1\, ((\pi_2\, x_O)\, (\pi_1\, y, x_1)), \iota_2\, ((\pi_2\, x_I)\, x_2)),
$$

which has type $\mathbf{1} \times (\mathbf{1} + \mathbf{1}) \to \mathbf{1}$.

Timing Evaluation Now that we have in our hands the λ-terms D_1, D_2, we can evaluate them to work out the timing table for CIRC. This timing table is parametric in the timing tables T_{OR}, T_{INV}, and T_{AND} for the components OR, INV, and AND, respectively, which must be substituted for the free variables x_O, x_I, and x_A. The timing table for CIRC then is uniquely determined by three numbers $\delta_{\text{CIRC}\uparrow}$, $\delta^1_{\text{CIRC}\downarrow}$, and $\delta^2_{\text{CIRC}\downarrow}$ which are computed as follows:

$$
\begin{array}{rcl}
\delta_{\text{CIRC}\uparrow} & = & \pi_1\, [\![([D_1]_{x_A,x_I,x_O}\, \{T_{\text{OR}}/x_O\}\{T_{\text{INV}}/x_I\}\{T_{\text{AND}}/x_A\})\, (del(0), del(0))]\!] \\
\delta^1_{\text{CIRC}\downarrow} & = & \pi_1\, [\![([D_2]_{x_A,x_I,x_O}\, \{T_{\text{OR}}/x_O\}\{T_{\text{INV}}/x_I\}\{T_{\text{AND}}/x_A\})\, (del(0), \iota_1(del(0)))]\!] \\
\delta^2_{\text{CIRC}\downarrow} & = & \pi_1\, [\![([D_2]_{x_A,x_I,x_O}\, \{T_{\text{OR}}/x_O\}\{T_{\text{INV}}/x_I\}\{T_{\text{AND}}/x_A\})\, (del(0), \iota_2(del(0)))]\!].
\end{array}
$$

It is not difficult to compute these delays by unravelling the semantics in Sec. 6:

$$
\begin{array}{rcl}
\delta_{\text{CIRC}\uparrow} & = & max(\delta^1_{\text{OR}\uparrow}, \delta_{\text{INV}\uparrow}) + \delta_{\text{AND}\uparrow} \\
\delta^1_{\text{CIRC}\downarrow} & = & \delta_{\text{OR}\downarrow} + \delta^1_{\text{AND}\downarrow} \\
\delta^2_{\text{CIRC}\downarrow} & = & \delta_{\text{INV}\downarrow} + \delta^2_{\text{AND}\downarrow}.
\end{array}
$$

The delays are data-dependent and take account of what part of the functionality is achieved by which components and along which path through the circuit. For instance, the falling output transition triggered by $a = 0$ and $b = 0$ has delay $\delta^1_{\text{CIRC}\downarrow} = \delta_{\text{OR}\downarrow} + \delta^1_{\text{AND}\downarrow}$ since the falling edge in this case propagates through the OR and the *first* input of the AND (*cf.* Fig. 2). On the other hand, if the falling output transition is triggered by $a = 0$ and $b = 1$, then the falling edge travels through INV and the *second* input of AND, whence

$\delta^2_{\text{CIRC}\downarrow} = \delta_{\text{INV}\downarrow} + \delta^2_{\text{AND}\downarrow}$. Only for a rising output transition all three gates are used, and we get $\delta_{\text{CIRC}\uparrow} = max(\delta^1_{\text{OR}\uparrow}, \delta_{\text{INV}\uparrow}) + \delta_{\text{AND}\uparrow}$. Note that all three delays $\delta_{\text{CIRC}\uparrow}$, $\delta^1_{\text{CIRC}\downarrow}$, and $\delta^2_{\text{CIRC}\downarrow}$ may actually be smaller than the so-called *topological* delay through the circuit, *i.e.* the longest path.

8 Critical Discussion and Future Work

In this report we proposed an application of constructive theorem proving to the timing analysis of hardware. A concrete framework has been outlined based on a special semantic interpretation of intuitionistic propositional logic, combined with an appropriate stabilization semantics for proofs. Via the semantic interpretation intuitionistic logic is turned into an adequate means to reason about the functional behaviour of combinational circuits in the presence of propagation delays, oscillations, and glitches. We have shown, by way of an example, that data-dependent timing information for combinational circuits can be extracted using the proposed stabilization semantics of proofs. The usual Boolean steady-state analysis is contained as a special case, though the interplay with respect to timing extraction between the classical analysis (essentially the negated formulas) and the intuitionistic part needs further exploration.

We have implemented an experimental prototype system for the timing analysis of combinational circuits based on the ideas presented in this report. The code for the central part of the system, the LJ theorem prover, was obtained from Roy Dyckhoff. It implements a variant of his contraction-free system [3] with optimizations due to Torkel Franzén [5]. Among the other parts of the program are a lambda compiler for proof scripts, and a timing evaluator for lambda terms. All code is written in Prolog and implemented in BinProlog [15]. Though the applicability has been established in theory and small circuits have been analyzed, it is too early to assess the feasibility of the approach on realistic circuit sizes. The development of this system is still in its initial stages.

Given a circuit specification ϕ, every proof of ϕ gives rise to a uniform stabilization bound for ϕ. Hence, this bound will be a property not only of the circuit but also of a particular proof. This dependency needs further exploration. For instance, if a circuit has several alternative ways of achieving a particular function we may have different proofs and consequently different stabilization bounds. We conjecture that in these cases the actual delay of the circuit can be characterized as the *minimal* uniform stabilization bound for ϕ.

The theory of circuits defined as $\mathbf{ThC} = \{\, \phi \mid\, \models \phi \,\}$ is a deductively closed set of formulas, that deserves to be investigated in its own right. We know by Thm. 4.1 that LJ (over the atomic sentences $a = i$) is contained in **ThC**. It is not difficult to see that in fact **ThC** is properly contained between LJ and classical logic, whence it is a so-called *intermediate* logic. The axiom scheme KP of Kreisel and Putnam, $(\neg\phi \supset \psi_1 \vee \psi_2) \supset ((\neg\phi \supset \psi_1) \vee (\neg\phi \supset \psi_2))$, for instance, is contained in **ThC**, but not in LJ. It is beyond the scope of this report to discuss the issue of whether it is possible to give a complete axiomatization for **ThC**. This is left as an open problem.

We anticipate an extension of the timing interpretation to constructive first-order and higher-order logic, in particular for the Calculus of Constructions (CC). A related, but more general, approach to the extraction of (hardware) constraints based on the Lego prover [10] (which implements CC) has been presented already in [11, 12]. The timing interpretation discussed in this report is a special instance of this general idea, and might be implemented as an extension to the Lego tool.

References

[1] ACM. *International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, Intermar Hotel Malente, Germany, September 1993.

[2] A. G. Dragalin. *Mathematical Intuitionism. Introduction to Proof Theory.* American Mathematical Society, 1988.

[3] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *The Journal of Symbolic Logic*, 57(3):795–807, September 1992.

[4] M. Fairtlough and M. Mendler. An intuitionistic modal logic with applications to the formal verification of hardware. In *Computer Science Logic*, Kazimierz, Poland, September 1994. University of Warsawa.

[5] Torkel Franzén. Algorithmic aspects of intuitionistic propositional logic. Research Report SICS R87010, Swedish Institute of Computer Science, 1987.

[6] G. Gentzen. Untersuchungen über das Logische Schließen. *Math. Z.* , 39:176–210, 405–431, 1934–1935.

[7] J. Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types.* Cambridge University Press, 1989.

[8] C. T. Gray, W. Liu, and R. K. Cavin III. Exact timing analysis considering data dependent delays. [1].

[9] D. Gurr. *Semantic Frameworks for Complexity.* PhD thesis, Edinburgh University, Department of Computer Science, January 1991.

[10] Lego. The Lego proof assistant. `http://www.dcs.ed.ac.uk/packages/lego` on the World Wide Web, 1995.

[11] M. Mendler. Constrained proofs: a logic for dealing with behavioural constraints in formal hardware verification. In G. Jones and M. Sheeran, editors, *Workshop on Designing Correct Circuits.* Springer, 1991.

[12] M. Mendler. *A Modal Logic for Handling Behavioural Constraints in Formal Hardware Verification.* PhD thesis, Department of Computer Science, University of Edinburgh, ECS-LFCS-93-255, March 1993.

[13] M. Mendler. A timing refinement of intuitionistic proofs and its application to the timing analysis of combinational circuits. Technical Report MIP-95??, University of Passau, December 1995.

[14] E. Moggi. Computational lambda-calculus and monads. In *Proceedings LICS'89*, pages 14–23, June 1989.

[15] P. Tharau. BinProlog 3.45 User Guide. Departement d'Informatique, Université de Moncton, Canada, June 1995.