Software ENgineering

SEN

The Differential Calculus of Bitstreams

J.J.M.M. Rutten

REPORT SEN-E0403 MARCH, 10 2004

ISSN 1386-369X

The Differential Calculus of Bitstreams (extended abstract)

ABSTRACT
Using (stream) differential equations for definitions and coinduction for proofs, we define, analyse, and relate in a uniform way four different algebraic structures on the set 2^{ω} of bitstreams, motivating each of them in terms of the digital circuits they can describe.

2000 Mathematics Subject Classification: 68Q10; 68Q45; 68Q55; 68Q70; 68Q85
Keywords and Phrases: Coalgebra; Bisimulation; Coinduction; Stream; Stream Calculus; Input derivative; Digital Circuit

The Differential Calculus of Bitstreams (extended abstract)

J.J.M.M. Rutten

CWI and VUA, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract. Using (stream) differential equations for definitions and coinduction for proofs, we define, analyse, and relate in a uniform way four different algebraic structures on the set 2^ω of bitstreams, motivating each of them in terms of the digital circuits they can describe.

1 Introduction

The notion of input derivative of a formal language was introduced by Brzozowski [Brz64] and has since been used in various algorithms for the efficient construction of language accepting automata [Con71,BS86]. In [Rut98,Rut99], we explained the role of input derivatives in terms of the final coalgebra structure on the set of all formal languages (over a fixed alphabet); generalised it to formal power series over arbitrary semirings; and showed that input derivatives can be used both to define languages and power series, in terms of so-called behavioural differential equations, and to prove their equality, by coinduction. In [Rut01], we used this approach to develop a calculus of infinite sequences of real numbers. (See also [LS02,CP03] for recent related work on the use of input derivatives and coinduction.) In the present paper, we apply the above methodology to the set 2^ω of bitstreams (infinite sequences of 0's and 1's). Bitstreams can be viewed as a natural extension of the set of Booleans $\{0, 1\}$ (over time), which one needs to describe the behaviour of *sequential* circuits, that is, digital circuits with feedback. We define in a uniform manner various operators on 2^ω by means of (stream) differential equations, and prove many properties about them by coinduction. These operators can be organised in four groups, each of which constitutes a different algebraic structure on 2^ω: a Boolean algebra, a Kleene algebra, and two integral domains. We shall motivate the relevance of each of these algebras in terms of the digital circuits that they can describe. Moreover, we shall prove various *mixed* laws that relate operators belonging to different algebraic structures. There does not seem to exist much literature on the systematic study of 2^ω in the context of digital circuits. A notable exception is the work by Vuillemin [Vui94,Vui00,Vui03], who analyses three of the four algebraic structures mentioned above, from the perspective of the 2-adic numbers, and studies algorithms for the construction of digital circuits from algebraic descriptions. Apart from the latter point, which we do not address here, the present treatment of bitstreams extends the results of Vuillemin by the following main contributions: (a) the uniform definition (by means of one large system of differential equations) of all the (twelve) operators of the four algebraic structures (Theorem 3), without the need of additional (ultrametric or) topological structure on 2^ω; (b) the formulation and systematic proof (by coinduction) of various existing and new mixed laws; (c) the introduction and study of the Kleene algebra structure on 2^ω, which turns out to be a basic and useful model for sequential circuits; and (d) the complete characterisation of three classes of linear circuits in terms of three corresponding notions of rational streams (Theorem 10).

2 The final coalgebra of bitstreams

We introduce the set of bitstreams, the operations of initial value and stream derivative, and the proof and definition principles of coinduction.

Let $2 = \{0, 1\}$ and $\mathbb{N} = \{0, 1, 2, \ldots\}$. We define the set of bitstreams (streams for short) by $2^\omega = \{\sigma \mid \sigma : \mathbb{N} \to 2\}$. Individual bitstreams will be denoted by $\sigma = (\sigma(0), \sigma(1), \sigma(2), \ldots)$ (sometimes also written as $(\sigma_0, \sigma_1, \sigma_2, \ldots)$). The following convention will be useful: we include the set 2 into the set 2^ω by putting $[0] = (0, 0, 0, \ldots)$ and $[1] = (1, 0, 0, 0, \ldots)$. For a stream $\sigma \in 2^\omega$, we call $\sigma(0)$ the *initial value* and we define the *(stream) derivative* of σ by $\sigma' = (\sigma(1), \sigma(2), \sigma(3), \ldots)$. (In computer science, initial value and derivative of a stream are better known as *head* and *tail*.) Initial value and derivative define in fact two functions $(-)(0) : 2^\omega \to 2$ and $(-)' : 2^\omega \to 2^\omega$, which turn the set 2^ω into a *coalgebra* $(2^\omega, (-)(0), (-)')$. This coalgebra is *final* in the set of all (similar such) coalgebras, in the following sense: For every pair (S, o, t) consisting of a set S and functions $o : S \to 2$ and $t : S \to S$, there exists a unique function $h : S \to 2^\omega$ (called a homomorphism of coalgebras) such that, for all $s \in S$, $o(s) = h(s)(0)$ and $(h(s))' = h(t(s))$. (A proof is easy: take for h the function given by $h(s) = (o(s), o(t(s)), o(t(t(s))), \ldots)$, for all $s \in S$, and show that it is the only possible choice.) The theory of coalgebra will play no explicit role in the present paper (see [JR97,Rut01] for more information on coalgebra). We have mentioned it nonetheless, because it shows that the operations of initial value and derivative are universal (precisely in the sense that finality is called a universal property in category theory). Moreover, it is the basis for the proof and definition principles of coinduction, which we introduce next.

Coinductive proofs will be formulated in terms of the following notion. A relation $R \subseteq 2^\omega \times 2^\omega$ is called a *(stream) bisimulation* if, for all $\langle\sigma, \tau\rangle \in R$, $\sigma(0) = \tau(0)$ and $\langle\sigma', \tau'\rangle \in R$. In contrast to the simplicity of its proof (omitted here), the following theorem, called the *coinduction proof principle*, will turn out to be surprisingly powerful.

Theorem 1. *For all $\sigma, \tau \in 2^\omega$: if there is a bisimulation $R \subseteq 2^\omega \times 2^\omega$ with $\langle\sigma, \tau\rangle \in R$, then $\sigma = \tau$.*

Thus in order to prove the equality of two bitstreams, it suffices to construct a bisimulation that relates them. See [Rut01,Rut03] for many examples of proofs by coinduction in the setting of streams of real numbers. In the present paper, most equalities are presented without proof, and only a few examples of proofs by coinduction have been included in the appendix, for lack of space.

Next we show how we can define constant bitstreams and functions on bitstreams, much as in mathematical calculus, by *(stream) differential equations*, which specify their initial value and derivative. For instance, $\gamma' = \gamma$ and $\gamma(0) = 1$ defines the constant stream $\gamma = (1, 1, 1, \ldots)$; and the two equations $f(\sigma, \tau)' = g(\sigma', \tau)$, $f(\sigma, \tau)(0) = \sigma(0)$ and $g(\sigma, \tau)' = f(\sigma, \tau')$ $g(\sigma, \tau)(0) = \tau(0)$, define the function $f(\sigma, \tau) = (\sigma(0), \tau(0), \sigma(1), \tau(1), \ldots)$. The following theorem asserts the existence of a unique solution for a large class of differential equations. It may well be skipped at first reading. The only potential difficulty lies in its generality. For all the concrete differential equations that will follow in this paper, the reader should have no difficulty in proving by elementary means that they have a unique solution.

Theorem 2. *Let Σ be a set of function symbols. Consider the following system of* (stream) differential equations*, one for every $g \in \Sigma$ with arity $r \in \mathbb{N}$:*

$$g(\sigma_1, \ldots, \sigma_r)' = T(\sigma_1, \ldots, \sigma_r, \sigma_1', \ldots, \sigma_r', [\sigma_1(0)], \ldots, [\sigma_r(0)])$$

$$g(\sigma_1, \ldots, \sigma_r)(0) = \phi(\sigma_1(0), \ldots, \sigma_r(0))$$

where: (i) $\sigma_1, \ldots, \sigma_r \in 2^\omega$; (ii) T is an arbitrary composition of function symbols from Σ (possibly including g itself), applied to (a subset of) the arguments that are listed (recall that $[a] = (a, 0, 0, 0, \ldots)$ for $a \in 2$); and (iii) $\phi : 2^r \to 2$ is an arbitrary function (or a constant in case $r = 0$). Then there is a unique family of functions $(g : (2^\omega)^r \to 2^\omega)_{g \in \Sigma}$ satisfying the system of equations above.

A sketch of the proof is included in the appendix. The above result may seem not sufficiently constructive as it only states the existence of a solution without giving an explicit construction for it. However, since (functions on) streams are completely determined by their derivative and initial value, which *are* specified by the differential equations, properties of such solutions can be easily established using the coinduction proof principle. Further note that Theorem 2 gives us a very general definition principle, which is of a syntactic nature: it suffices to look at the syntactic format of the differential equations to be sure that they have a unique solution. Theorem 2 is sufficient for our present purposes but can be generalised in many different ways. For instance, the term T above could also be applied to higher-order derivatives of the arguments of g (as in $\mathit{even}(\sigma)' = \mathit{even}(\sigma'')$, $\mathit{even}(\sigma)(0) = \sigma(0)$, which defines $\mathit{even}(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \ldots)$).

3 The differential calculus of bitstreams

The following theorem introduces the main operators on bitstreams in one large system of differential equations. Their definition is based on the usual Boolean operators *or*, *and*, and *negation*, which are defined, for all $a, b \in 2 = \{0, 1\}$, by $a + b = \max\{a, b\}$, $a \cdot b = \min\{a, b\}$, $\overline{a} = 1 - a$. We shall moreover use the operation of *addition modulo-2* (aka exclusive or): $a \oplus b = (a \cdot \overline{b}) + (\overline{a} \cdot b)$.

Theorem 3. *There exist unique operators* $+, \wedge, \ldots, 1/\!\!/(-)$ *defined by the following system of differential equations. Below we write* σ_0 *and* τ_0 *for* $\sigma(0)$ *and* $\tau(0)$ *and we use the constants* $[a] = (a, 0, 0, \ldots)$*, for* $a \in 2$*.*

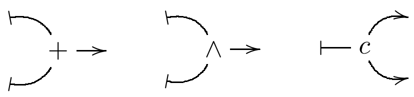
derivative:	*initial value:*	*name:*
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma_0 + \tau_0$	*or*
$(\sigma \wedge \tau)' = \sigma' \wedge \tau'$	$(\sigma \wedge \tau)(0) = \sigma_0 \cdot \tau_0$	*and*
$(\neg\sigma)' = \neg(\sigma')$	$(\neg\sigma)(0) = \overline{\sigma_0}$	*negation*
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma_0 + \tau_0$	*sum (= or)*
$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma_0] \times \tau')$	$(\sigma \times \tau)(0) = \sigma_0 \cdot \tau_0$	*(convolution) product*
$(\sigma^*)' = \sigma' \times \sigma^*$	$(\sigma^*)(0) = 1$	*(Kleene) star*
$(\sigma \oplus \tau)' = \sigma' \oplus \tau'$	$(\sigma \oplus \tau)(0) = \sigma_0 \oplus \tau_0$	*sum modulo-2*
$(\sigma \otimes \tau)' = (\sigma' \otimes \tau) \oplus ([\sigma_0] \otimes \tau')$	$(\sigma \otimes \tau)(0) = \sigma_0 \cdot \tau_0$	*product modulo-2*
$(1 \oslash \sigma)' = \sigma' \otimes (1 \oslash \sigma)$	$(1 \oslash \sigma)(0) = 1$	*inverse mod-2* $(\sigma_0 = 1)$
$(\sigma \mathbin{+\!\!\!+} \tau)' = (\sigma' \mathbin{+\!\!\!+} \tau') \mathbin{+\!\!\!+} [\sigma_0 \cdot \tau_0]$	$(\sigma \mathbin{+\!\!\!+} \tau)(0) = \sigma_0 \oplus \tau_0$	*2-adic sum*
$(\sigma \mathbin{\times\!\!\!\times} \tau)' = (\sigma' \mathbin{\times\!\!\!\times} \tau) \mathbin{+\!\!\!+} ([\sigma_0] \mathbin{\times\!\!\!\times} \tau')$	$(\sigma \mathbin{\times\!\!\!\times} \tau)(0) = \sigma_0 \cdot \tau_0$	*2-adic product*
$(1/\!\!/\sigma)' = (-\sigma') \mathbin{\times\!\!\!\times} (1/\!\!/\sigma)$	$(1/\!\!/\sigma)(0) = 1$	*2-adic inverse* $(\sigma_0 = 1)$

(Note that we are using the symbol $+$ both for *or* on the set 2 and for *sum* on the set 2^ω; similarly for $\oplus$.) The proof is an immediate consequence of Theorem 2. We shall see examples of how to compute with the above operators later. Let us emphasize the algorithmic aspect of such stream differential equations. They allow us to compute the respective elements of the streams they define, by repeatedly computing derivatives and taking their initial values. The operators above have been organised in four groups of three, corresponding to the four different algebraic structures on 2^ω considered in the present paper, namely: Boolean algebra, Kleene algebra, the integral domain corresponding to sum and product modulo-2, and the integral domain corresponding to the 2-adic operators. In the subsequent sections, we will treat each of these structures in detail by: motivating the above operators by means of digital circuits that can be described by them; characterising the operators (sometimes in terms of additionally defined operators) by stating their basic properties; and, most importantly, by formulating mixed laws that relate operators from different algebraic structures.

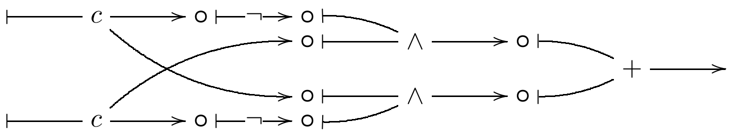
4 Boolean algebra

The structure $(2^\omega, +, \wedge, \neg, [0], \langle 1 \rangle)$ consists of the set of bitstreams, the operations of *or*, *and*, *negation*, and the constants $[0] = (0, 0, 0, \ldots)$ and $\langle 1 \rangle = (1, 1, 1, \ldots)$. It inherits its Boolean algebra structure from the Boolean operators on $2 = \{0, 1\}$ (mentioned at the beginning of Section 3) in an elementwise fashion: $(\sigma + \tau)(n) = \sigma(n) + \tau(n)$, $(\sigma \wedge \tau)(n) = \sigma(n) \cdot \tau(n)$, $(\neg\sigma)(n) = \overline{\sigma(n)}$, for all $\sigma, \tau \in 2^\omega$, $n \geq 0$. (We use the same symbol $+$ for both Booleans and bitstreams; the context will always make clear which.) The constants are the neutral elements for *or* and *and*: $[0] + \sigma = \sigma$ and $\langle 1 \rangle \wedge \sigma = \sigma$, and we have the familiar laws from Boolean algebra such as $\neg(\sigma + \tau) = \neg\sigma \wedge \neg\tau$, $\sigma + \sigma = \sigma$, and the like. The Boolean operators on 2^ω are sufficient to describe the behaviour of so-called

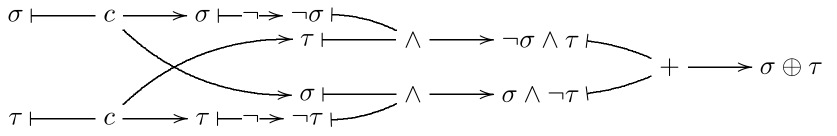
Boolean or *logical circuits*, which are built using *or*-, *and*-, and *not*-gates, with the help of a *copier*, and without allowing delay or feedback. We denote the not-gate by $\vdash\!\neg\!\rightarrow$ and the other three gates by



Note that we denote input ends by $\vdash$ and output ends by $\rightarrow$. Composition of gates will be explicitly denoted by the symbol for function composition $\circ$, as in the following example:



The explicit denotation of copiers and composition may seem a bit pedantic but will allow us to compute the input/output behaviour of circuits by systematically replacing the $\circ$ symbols by corresponding streams, as follows. For all input streams σ and τ, we have



where the output is defined by $\sigma \oplus \tau = (\neg\sigma \wedge \tau) + (\sigma \wedge \neg\tau)$. As we shall see, this method works in general, notably also for circuits with feedback.

5 Kleene algebra

The structure $(2^\omega, +, \times, (-)^*, [0], [1])$ will allow us to handle the first examples of so-called *sequential* circuits. Similarly to Boolean circuits, sequential circuits are built from and-, or-, and not-gates, but in addition they may contain registers (to be introduced below) and feedback. The structure $(2^\omega, +, \times, (-)^*, [0], [1])$ consists of the set of bitstreams together with the operations of *sum*, *convolution product*, *Kleene star* (and the constants $[0] = (0, 0, 0, \ldots)$ and $[1] = (1, 0, 0, 0, \ldots)$) defined by the differential equations in Section 3. The following properties follow from the definitions of the operators: Without the star, the above structure is a *semi-ring* [BR88] (which is more or less a ring without subtraction): sum is commutative, sum and product are associative and have neutral elements [0] and [1]: $\sigma + [0] = \sigma$, $\sigma \times [0] = [0]$, $\sigma \times [1] = \sigma$, and product distributes over sum in the usual way. This semiring is moreover idempotent and commutative: $\sigma + \sigma = \sigma$ and $\sigma \times \tau = \tau \times \sigma$. These two identities are an immediate consequence of the elementwise characterisations of sum and product:

$$(\sigma + \tau)(n) = \sigma(n) + \tau(n)$$

$$(\sigma \times \tau)(n) = (\sigma(0) \cdot \tau(n)) + \, (\sigma(1) \cdot \tau(n-1)) + \, \cdots + \, (\sigma(n) \cdot \tau(0))$$

which follow from the definitions. The presence of star makes of the above structure moreover a *Kleene algebra* [Koz94], satisfying the following laws, the first of which is particularly simple because of the idempotency of sum and the commutativity of product: for all $\sigma, \tau, \rho \in 2^\omega$,

$$(\sigma + \tau)^* \, = \sigma^* \times \tau^* \tag{1}$$

$$\sigma^* \, = [1] + (\sigma \times \sigma^*) \tag{2}$$

$$\sigma = (\rho \times \sigma) + \tau \Rightarrow \sigma = \rho^* \times \tau \tag{3}$$

where the latter law only holds if $\rho(0) = 0$. The following constant will help us in revealing much additional structure of 2^ω; moreover, it will be the key in the semantics of the delay gate that will be introduced shortly. We define:

$$X = (0, 1, 0, 0, 0, \ldots)$$

We have $X(0) = 0$ and $X' = [1]$. If we define $X^0 = [1]$ and $X^{n+1} = X \times X^n$, then $X^2 = (0, 0, 1, 0, 0, 0, \ldots)$, $X^3 = (0, 0, 0, 1, 0, 0, 0, \ldots)$, and so on, and $(X^{n+1})' = X^n$ (all $n \geq 0$). More generally, we have $X \times \sigma = (0, \sigma(0), \sigma(1), \sigma(2), \ldots)$ and $(X \times \sigma)' = \sigma$, making of $X \times (-)$ a kind of inverse to the operation of stream derivative. More precisely, there is the *fundamental theorem* of stream calculus: for all $\sigma \in 2^\omega$,

$$\sigma = [\sigma(0)] + \, (X \times \sigma') \tag{4}$$

Using the operation of (countably) infinite sum (defined for streams $(\tau_n)_{n\geq 0}$ by $(\sum \tau_n)' = \sum \tau_n$, $(\sum \tau_n)(0) = \max\{\tau_n(0) \mid n \geq 0\}$), we also have:

$$\sigma = [\sigma(0)] + \, ([\sigma(1)] \times X) + \, ([\sigma(2)] \times X^2) + \, \cdots = \sum [\sigma(n)] \times X^n$$

This formula shows that streams σ are formal power series in one formal variable (namely, the constant X).

The Kleene algebra of the present section can also be understood in formal language theoretical terms. We can associate with any stream σ the formal language $\{X^n \mid \sigma(n) = 1\}$, that is, a set of finite words over the one-letter alphabet X. The constant [0] corresponds to the empty language and [1] to the language $\{\varepsilon\}$ containing the empty word $X^0 = \varepsilon$; the operations of sum, product, and star correspond to the familiar operations of formal language theory: union, concatenation, and Kleene star. This correspondence will play no role in the remainder of this paper.

Examples 4 **(i)** For any $n \geq 0$ and $a_0, \ldots, a_n \in 2$, there is the *polynomial* stream $[a_0] + ([a_1] \times X) + \ldots + ([a_n] \times X^n) = (a_0, a_1, \ldots, a_n, 0, 0, 0, \ldots)$. Let

$$\langle a_0 a_1 \cdots a_n \rangle = (\, [a_0] + ([a_1] \times X) + \ldots + ([a_n] \times X^n) \,) \times \, (X^{n+1})^*$$

It is easy to show that $\langle a_0 a_1 \cdots a_n \rangle' = \langle a_1 \cdots a_n a_0 \rangle$, whence $\langle a_0 a_1 \cdots a_n \rangle = (a_0, a_1, \ldots, a_n, a_0, a_1, \ldots, a_n, \ldots)$. For instance, we have $\langle 1 \rangle = X^* = (1, 1, 1, \ldots)$

(which is to be distinguished from $[1] = (1, 0, 0, 0, \ldots)$), and $\langle 010 \rangle = X \times (X^3)^* = (0, 1, 0, 0, 1, 0, 0, 1, 0, \ldots)$.
(ii) We have $X^* \times \sigma = (\sigma(0), \sigma(0) + \sigma(1), \sigma(0) + \sigma(1) + \sigma(2), \ldots)$, for all $\sigma \in 2^\omega$.
(iii) For $\sigma, \tau \in 2^\omega$ with $\tau(0) = 0$, we define σ *applied to* τ by the following differential equation: $(\sigma(\tau))' = \sigma'(\tau) \times \tau'$, $(\sigma(\tau))(0) = \sigma(0)$. It follows that $\sigma(X^2) = (\sigma(0), 0, \sigma(1), 0, \sigma(2), 0 \ldots)$. One can now define the function *zip* by $\mathit{zip}(\sigma, \tau) = \sigma(X^2) + (X \times \tau(X^2)) = (\sigma(0), \tau(0), \sigma(1), \tau(1), \sigma(2), \tau(2), \ldots)$. □

We are now in a position to introduce a new basic gate: the *register* or *delay* gate $\vdash\!\xrightarrow{R}$, which acts as a one-element memory cell with initial value 0. Formally, it has the following behaviour:

$$\sigma \;\vdash\!\xrightarrow{R}\; (0, \sigma(0), \sigma(1), \sigma(2), \ldots) = X \times \sigma$$

Registers are typically used to 'guard' feedback loops, ensuring that they have a well-defined behaviour, as in the following example:

$$\begin{tikzcd}
& \rho_1 \arrow[d] & & \rho_2 \arrow[ll, "R"'] & \\
\sigma \arrow[r] & + \arrow[r] & \rho_3 \arrow[r] & c \arrow[r] \arrow[u] & \tau
\end{tikzcd}$$

In order to determine the output τ as a function of the input σ, we have included intermediate streams ρ_1, ρ_2, ρ_3. We find: $\tau = \rho_3 = \rho_2$, $\rho_3 = \sigma + \rho_1$, $\rho_1 = X \times \rho_2$, which implies $\tau = (X \times \tau) + \sigma$. By (3) and Examples 4**(ii)**,

$$\tau = X^* \times \sigma = (\sigma(0), \sigma(0) + \sigma(1), \sigma(0) + \sigma(1) + \sigma(2), \ldots)$$

Mixed Laws 5 The following laws combine operators from Boolean and Kleene algebra: $\neg[0] = \langle 1 \rangle$, $\neg[1] = X \times \langle 1 \rangle$, $\neg X = [1] + (X^2 \times \langle 1 \rangle)$, $\sigma \wedge [1] = [\sigma(0)]$, $\sigma \wedge \neg[1] = X \times \sigma'$, $\neg(X \times \sigma) = [1] + (X \times \neg\sigma)$, $\neg\langle a_0 a_1 \cdots a_n \rangle = \langle \overline{a_0}\, \overline{a_1} \cdots \overline{a_n} \rangle$. □

Here is another example of a circuit, with feedback through a not-gate:

$$\begin{tikzcd}
& \rho_1 \arrow[dl, "R"'] & & \\
\rho_2 \arrow[dr, "\neg"'] & & c \arrow[ul] \arrow[r] & \tau \\
& \rho_3 \arrow[ur] & &
\end{tikzcd}$$

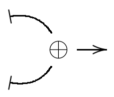
Note that even though this circuit has no inputs, it produces an output stream τ because of the presence of the register R that contains (by definition) an initial value 0. Formally: $\tau = \rho_3 = \neg\rho_2 = \neg(X \times \rho_1) = \neg(X \times \tau)$. This implies $\tau = \neg(X \times \tau) = [1] + (X \times \neg\tau)$ (by Mixed Laws 5) $= [1] + (X \times \neg(\neg(X \times \tau))) = [1] + (X^2 \times \tau)$. Using (3), we find $\tau = (X^2)^* = \langle 10 \rangle = (1, 0, 1, 0, 1, 0, \ldots)$. Similarly, one can compute that the next circuit:

$$\begin{tikzcd}
& \circ \arrow[d] & \circ \arrow[l, "\neg"'] & & \circ \arrow[ll, "R"'] & \circ \arrow[l, "\neg"'] & \\
\sigma \arrow[r] & \wedge \arrow[rr] & & \circ \arrow[rr] & & c \arrow[r] \arrow[u] & \tau
\end{tikzcd}$$

produces for every input σ an output τ satisfying $\tau = \sigma \wedge ([1] + (X \times \tau))$. This implies $\neg\tau = \neg\sigma + \neg([1] + (X \times \tau)) = \neg\sigma + (X \times \neg\tau)$. Thus $\neg\tau = X^* \times \neg\sigma$, by (3), whence $\tau = \neg(X^* \times \neg\sigma) = (\sigma(0), \sigma(0) \cdot \sigma(1), \sigma(0) \cdot \sigma(1) \cdot \sigma(2), \ldots)$.

6 Sum and product modulo-2

Recall the Boolean circuit at the end of Section 4 that produces the stream $\sigma \oplus \tau = (\neg\sigma \wedge \tau) + (\sigma \wedge \neg\tau)$ for inputs $\sigma, \tau \in 2^\omega$, and let the entire circuit be denoted by the following shorthand:



In the present section, we shall study the operator on streams $\oplus$ in its own right. It forms the basis for yet another interesting algebraic structure on 2^ω, with which we can give a concise description of the behaviour of sequential circuits that contain the above $\oplus$-gate. We consider $(2^\omega, \oplus, \ominus, \otimes, \oslash, [0], [1])$ consisting of the set of bitstreams with the operations of sum, minus, product and inverse *modulo-2* (and the constants [0] and [1]), all of which are defined by the differential equations in Section 3, of which the following properties are an immediate consequence. The above structure is a ring in which sum is idempotent and product is commutative, where [0] and [1] are the neutral elements for sum and product, and where the operation of minus is the identity, since $\sigma \oplus \sigma = 0$, for all $\sigma \in 2^\omega$. The structure is moreover an integral domain (that is, it has no zero-divisors) since $\sigma \neq [0]$ and $\tau \neq [0]$ imply $\sigma \otimes \tau \neq [0]$. Furthermore, the operation of $1 \oslash \sigma$ acts as an inverse to product: $\sigma \otimes (1 \oslash \sigma) = [1]$ for all σ with $\sigma(0) = 1$. We shall write $\sigma \oslash \tau$ for $\sigma \otimes (1 \oslash \tau)$. There are the following elementwise characterisations of $\oplus$ and $\otimes$, which are similar to those of $+$ and $\times$ in Section 5: $(\sigma \oplus \tau)(n) = \sigma(n) \oplus \tau(n)$ and $(\sigma \otimes \tau)(n) = (\sigma(0) \cdot \tau(n)) \oplus (\sigma(1) \cdot \tau(n-1)) \oplus \cdots \oplus (\sigma(n) \cdot \tau(0))$ (where we use $\oplus$ both for streams and for Booleans; the latter was introduced at the beginning of Section 3). For $\oslash$ we have the usual properties (such as $(1 \oslash \sigma) \otimes (1 \oslash \tau) = 1 \oslash (\sigma \otimes \tau)$). There is also the following law:

$$1 \oslash (1 \oplus (X \otimes \sigma)) = [1] \oplus (X \otimes \sigma) \oplus ((X \otimes \sigma) \otimes (X \otimes \sigma)) \oplus \cdots$$

where the infinite sum is defined similarly to the infinite sum of Section 5.

Mixed Laws 6 The following laws combine operators from the present and the previous sections: $[\sigma(0)] \oplus (X \otimes \sigma') = \sigma = [\sigma(0)] + (X \times \sigma')$. Note that this implies that the notation X^n and the notion of polynomial stream (Examples 4(i)) are unambiguous. Also: $\sigma \otimes \sigma = [\sigma(0)] + ([\sigma(1)] \times X^2) + ([\sigma(2)] \times X^4) + \cdots = \sigma(X^2)$ (with the latter as defined in Examples 4(iii)); $1 \oslash ([1] + X^{n+1}) = (X^{n+1})^*$; $1 \oslash ([1] + X + X^2 + \cdots + X^n) = ([1] + X) \oslash ([1] + X^{n+1})$; $\langle a_0 a_1 \cdots a_n \rangle = ([a_0] + ([a_1] \times X) + \ldots + ([a_n] \times X^n)) \oslash (1 + X^{n+1})$. Also, for polynomial streams σ and τ of degree at most $n \geq 0$,

$$(\sigma \oslash ([1] + X^{n+1})) \wedge (\tau \oslash ([1] + X^{n+1})) = (\sigma \wedge \tau) \oslash ([1] + X^{n+1})$$

Also, $\neg\sigma = \sigma \oplus (1 \oslash (1 \oplus X))$. $\square$

Examples 7 **(i)**: $1 \oslash ([1] \oplus X) = (1, 1, 1, \ldots) = X^*$, and $1 \oslash ([1] \oplus X \oplus X^2) = (1, 1, 0, 1, 1, 0, 1, 1, 0, \ldots) == \langle 110 \rangle = ([1] + X) \times (X^3)^*$.
(ii) We have $(1 \oslash (1 \oplus X)) \otimes \sigma = (\sigma(0), \sigma(0) \oplus \sigma(1), \sigma(0) \oplus \sigma(1) \oplus \sigma(2), \ldots)$, for all $\sigma \in 2^\omega$.

The following example combines feedback with addition modulo-2:

$$\begin{tikzcd}
 & \rho_1 \arrow[d, dash] & & \rho_2 \arrow[ll, maps to, "R" description] & \\
\sigma \arrow[r, maps to] & \oplus \arrow[r] & \rho_3 \arrow[r, maps to] & c \arrow[u] \arrow[r] & \tau
\end{tikzcd}$$

It is a variation of the first circuit in Section 5 in which we have replaced the sum (or) gate by the $\oplus$-gate, introduced at the beginning of this section. We compute the output τ as a function of the input σ, again with the help of intermediate streams ρ_1, ρ_2, ρ_3, as follows: $\tau = \rho_3 = \rho_2$, $\rho_3 = \sigma \oplus \rho_1$, $\rho_1 = X \times \rho_2 = X \otimes \rho_2$, which implies $\tau = (X \otimes \tau) \oplus \sigma$ and $([1] \oplus X) \otimes \tau = \sigma$. Compared to Section 5, where a similar equation was solved using the law (3), we can now more simply divide both sides by $[1] \oplus X$, yielding $\tau = (1 \oslash ([1] \oplus X)) \otimes \sigma$. Using Examples 7**(ii)**, we find $\tau = (\sigma(0), \sigma(0) \oplus \sigma(1), \sigma(0) \oplus \sigma(1) \oplus \sigma(2), \ldots)$. The above circuit is maybe more interesting than it seems at first sight. It is in fact a so-called T flip-flop (T for trigger) [Koh78] or, equivalently, a counter modulo-2: at any moment n in time, the output signal $\tau(n)$ is equal to the total number of 1 signals that have been received thus far, modulo-2.

7 The 2-adic operators

The operator of 2-adic sum can be easily explained in terms of a well-known circuit that implements it. Let H denote the familiar *half-adder* circuit that can be constructed out of the basic gates. It produces, for inputs σ and τ,

$$\begin{tikzcd}
\sigma \arrow[dr, maps to, no head, bend left] & & \tau \arrow[dl, maps to, no head, bend right] \\
 & H \arrow[dl, bend right] \arrow[r] & \sigma \wedge \tau \\
\sigma \oplus \tau & &
\end{tikzcd}$$

A (sequential) binary adder can now be constructed as follows:

$$\begin{tikzcd}
\sigma \arrow[dr, maps to, no head, bend left] & & \tau \arrow[dl, maps to, no head, bend right] & & \\
 & H \arrow[dl, bend right] \arrow[rr] & & \circ \arrow[dr, maps to, no head, bend left] & \\
\circ \arrow[dr, maps to, no head, bend left] & & \circ \arrow[dl, maps to, no head, bend right] & \circ \arrow[l, maps to, "R" description] & + \arrow[l] \\
 & H \arrow[dl, bend right] \arrow[rr] & & \circ \arrow[ur, maps to, no head, bend right] & \\
A(\sigma, \tau) & & & &
\end{tikzcd}$$

The output of this circuit, denoted for all $\sigma, \tau \in 2^\omega$ by $A(\sigma, \tau)$, is precisely $\sigma + \tau$:

$$\sigma + \tau = A(\sigma, \tau) \tag{5}$$

(See the appendix for a proof by coinduction.) We can now understand the defining differential equation for $\sigma \mathbin{+\!\!\!+} \tau$ from the table in Section 3: The initial value is given by $\sigma_0 \oplus \tau_0$, the sum of the initial values modulo 2; and the derivative consists of the (2-adic) sum of the derivatives σ' and τ' to which the *carry* $[\sigma_0 \cdot \tau_0]$ is added. The equations for 2-adic operators of product and inverse pretty much follow the pattern of the corresponding equations for the operators modulo-2. The structure $(2^\omega, \mathbin{+\!\!\!+}, -, \divideontimes, \mathbin{/\!\!/}, [0], [1])$ is again a commutative ring and integral domain, with familiar properties such as $\sigma \divideontimes (\tau \mathbin{+\!\!\!+} \rho) = (\sigma \divideontimes \tau) \mathbin{+\!\!\!+} (\sigma \divideontimes \rho)$. Sum, however, is no longer idempotent and we have, for all $\sigma \in 2^\omega$,

$$\sigma \mathbin{+\!\!\!+} ([1] \mathbin{+\!\!\!+} \neg\sigma) = [0] \tag{6}$$

For the 2-adic inverse, we have

$$1 \mathbin{/\!\!/} ([1] - (X \divideontimes \sigma)) = [1] \mathbin{+\!\!\!+} (X \divideontimes \sigma) \mathbin{+\!\!\!+} ((X \divideontimes \sigma) \divideontimes (X \divideontimes \sigma)) \mathbin{+\!\!\!+} \cdots$$

where the infinite sum is defined similarly to the infinite sum of Section 5.

Mixed Laws 8 The following laws combine operators from the present and the previous sections: $[\sigma(0)] \mathbin{+\!\!\!+} (X \divideontimes \sigma') = \sigma = [\sigma(0)] + (X \times \sigma') = [\sigma(0)] \oplus (X \otimes \sigma')$. Note that this implies that the notation X^n and the notion of polynomial stream (Examples 4(i)) remain unambiguous. Also: $-\sigma = [1] \mathbin{+\!\!\!+} \neg\sigma$; $(-\sigma)' = (-\sigma) \mathbin{+\!\!\!+} (\sigma')$; $1 \mathbin{/\!\!/} ([1] - X) = -[1]$; $\sigma \mathbin{+\!\!\!+} \tau = (\sigma + \tau) \mathbin{+\!\!\!+} (\sigma \wedge \tau)$; $\sigma \mathbin{+\!\!\!+} \tau = (\sigma \oplus \tau) \mathbin{+\!\!\!+} (X \times (\sigma \wedge \tau))$. Note that the latter implies $\sigma \mathbin{+\!\!\!+} \sigma = X \times \sigma$. □

Bitstreams together with the 2-adic operators are known as the *2-adic integers* [Kob84,Vui94]. With the present calculus, we can easily perform calculations on them, as the following examples illustrate.

Examples 9 Let $\sigma = [1] + X + X^2 = (1, 1, 1, 0, 0, 0, \ldots)$ and $\tau = X + X^3 = (0, 1, 0, 1, 0, 0, 0, \ldots)$ (which are binary representations of the integers 7 and 10). We have $\sigma \mathbin{+\!\!\!+} \tau = ([1] + X + X^2) \mathbin{+\!\!\!+} (X + X^3) = [1] \mathbin{+\!\!\!+} X \mathbin{+\!\!\!+} X^2 \mathbin{+\!\!\!+} X \mathbin{+\!\!\!+} X^3 = [1] \mathbin{+\!\!\!+} (X \mathbin{+\!\!\!+} X) \mathbin{+\!\!\!+} X^2 \mathbin{+\!\!\!+} X^3 = [1] \mathbin{+\!\!\!+} (X^2 \mathbin{+\!\!\!+} X^2) \mathbin{+\!\!\!+} X^3 = [1] \mathbin{+\!\!\!+} (X^3 \mathbin{+\!\!\!+} X^3) = [1] \mathbin{+\!\!\!+} X^4$ $(= (1, 0, 0, 0, 1, 0, 0, 0, \ldots)$, the binary representation of 17). Also, we have $\sigma \divideontimes \tau = ([1] + X + X^2) \divideontimes (X + X^3) = (X + X^2 + X^3) \mathbin{+\!\!\!+} (X^3 + X^4 + X^5) = X + X^2 + X^6$ (which equals $(0, 1, 1, 0, 0, 0, 1, 0, 0, 0, \ldots)$, the binary representation of 70). Using $-[1] = (1, 1, 1, \ldots)$ and the differential equations of the 2-adic operators, we also have, for instance, $1 \mathbin{/\!\!/} ([1] + X) = (1, 1, 0, 1, 0, 1, 0, \ldots)$ (which corresponds to a binary representation of the number 1/3). □

Here is a basic illustration of a circuit with feedback involving the binary adder circuit mentioned above:

$$\begin{tikzcd}
 & \rho_1 \arrow[d, dash] & & \rho_2 \arrow[ll, "R"'] & \\
\sigma \arrow[r, dash] & \mathbin{+\!\!\!+} \arrow[r] & \rho_3 \arrow[r, dash] & c \arrow[u] \arrow[r] & \tau
\end{tikzcd}$$

In this circuit, which is again a variation of the first circuit in Section 5, we have used a $\mathbin{+\!\!\!+}$-gate as a shorthand for the (sequential) binary adder at the beginning of the present section. Computing its behaviour, we find $\tau = \rho_3 = \rho_2$, $\rho_3 = \sigma \mathbin{+\!\!\!+} \rho_1$, $\rho_1 = X \times \rho_2 = X \mathbin{\times\!\!\!\times} \rho_2$, which implies $\tau = (X \mathbin{\times\!\!\!\times} \tau) \mathbin{+\!\!\!+} \sigma$ and $\tau \mathbin{\times\!\!\!\times} ([1] - X) = \sigma$. Thus we find $\tau = (1 \mathbin{/\!\!/} ([1] - X)) \mathbin{\times\!\!\!\times} \sigma = -\sigma$, using the identity $1 \mathbin{/\!\!/} ([1] - X) = -[1]$ from Mixed Laws 8 above.

8 Rational streams and linear circuits

Let $s \in \{+, \oplus, \mathbin{+\!\!\!+}\}$ express that s is any of the three operators for sum and, similarly, let an s-gate be any of the three corresponding sum gates. We call a circuit *s-linear* if it is built out of s-gates, copiers, and registers, possibly using feedbacks (that pass as always through at least one register). Notably, a linear circuit does not contain and- or not-gates. Linear circuits can be characterised in terms of *rational* streams, which we define next (recall the notion of polynomial stream from Examples 4):

(i) $\rho \in 2^\omega$ is *$+$-rational* if ρ can be constructed out of the constant streams $[0]$ and $[1]$ by means of the operators $+$, $\times$ and $(-)^*$.
(ii) $\rho \in 2^\omega$ is *$\oplus$-rational* if $\rho = \pi \oslash \tau$ for polynomial streams π, τ (with $\tau(0) = 1$).
(iii) $\rho \in 2^\omega$ is *$\mathbin{+\!\!\!+}$-rational* if $\rho = \pi \mathbin{/\!\!/} \tau$ for polynomial streams π, τ (with $\tau(0) = 1$).

Theorem 10. *Let $s \in \{+, \oplus, \mathbin{+\!\!\!+}\}$. A finite s-linear circuit with one input and one output end implements a stream function $f : 2^\omega \to 2^\omega$ of the following form: there exists an s-rational stream $\rho \in 2^\omega$ such that, for all $\sigma \in 2^\omega$,*

$$f(\sigma) = \begin{cases} \rho \times \sigma & \text{if } s = + \\ \rho \otimes \sigma & \text{if } s = \oplus \\ \rho \mathbin{\times\!\!\!\times} \sigma & \text{if } s = \mathbin{+\!\!\!+} \end{cases}$$

Conversely, any such function can be implemented by a finite s-linear circuit.

□

The above result for $\oplus$ is well-known, although often presented in a semi-formal symbolic form (cf. [Koh78]). The other two cases seem to be new. The proof of this theorem consists of a generalisation of the treatment of the various concrete examples of linear circuits that we have seen so far and is omitted for lack of space. (The proof of part (iii) is to be included in paper in preparation, on arithmetic on bitstreams.)

9 Discussion

We briefly mention some points for further research. We have presented and proved several mixed laws in a uniform way, and have illustrated how they can be used to obtain closed formulae describing the behaviour of various example circuits. We have obtained a systematic description of the behaviour of circuits only for the three classes of linear circuits, but not in general. More laws are needed in order to describe additional larger classes of circuits. Related to that are also the issues of expressiveness and of axiomatic characterisations of such classes. Another question goes back to the original motivation for the notion of (input) derivative (of languages), namely, the efficient construction of automata for languages. Similarly, one may ask whether derivatives can be used for a systematic and efficient construction of circuits implementing certain (operators on) bitstreams. We know how to do this only for linear circuits (the method is implicit in the omitted proof of the corresponding part of Theorem 10). We expect that the analysis of algebraic expressions in terms of their derivatives will lead to new uses of this method, which may also be useful for a further analysis and substantiation of the algorithms sketched in [Vui94,Vui03]. Finally, we intend to investigate how the present calculus of bitstreams can be of help in hardware specification and verification, and to compare it to other logical and algebraic approaches such as [Mel93] and [HNR03].

References

[BR88] J. Berstel and C. Reutenauer. *Rational series and their languages.* Springer-Verlag, 1988.

[Brz64] J.A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964.

[BS86] G. Berry and R. Sethi. From regular expressions to deterministic automata. *Theoretical Computer Science*, 48:117–126, 1986.

[Con71] J.H. Conway. *Regular algebra and finite machines.* Chapman and Hall, 1971.

[CP03] Hubie Chen and Riccardo Pucella. A coalgebraic approach to Kleene algebra with tests. In volume 82(1) of *ENTCS*. Elsevier, 2003.

[HNR03] E.C.R. Hehner, T.S. Norvell, and R.F.Paige. High-level circuit design. In *Programming Methodology*, chapter 18, pages 381–412. Springer, 2003.

[JR97] B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the EATCS*, 62:222–259, 1997. Available at URL: www.cwi.nl/∼janr.

[Kob84] N. Koblitz. *p-adic Numbers, p-adic Analysis, and Zeta-Functions*, volume 58 of *Graduate Texts in Mathematics.* Springer-Verlag, 1984.

[Koh78] Z. Kohavi. *Switching and finite automata theory.* McGraw-Hill, 1978.

[Koz94] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110:366–390, 1994.

[LS02] S. Lombardy and J. Sakarovitch. Derivatives of rational expressions with multiplicity. In LNCS 2420. Springer-Verlag, 2002.

[Mel93] T. Melham. *Higher Order Logic and Hardware Verification.* CUP, 1993.

[Rut98] J.J.M.M. Rutten. Automata and coinduction. In CONCUR'98, LNCS 1466, 1998.

[Rut99] J.J.M.M. Rutten. Automata, power series, and coinduction. In ICALP'99, LNCS 1644, 1999.

[Rut01] J.J.M.M. Rutten. Elements of stream calculus. In MFPS 2001, ENTCS 45, 2001.

[Rut03] J.J.M.M. Rutten. Behavioural differential equations. *TCS* 308, 2003.

[Vui94] J. Vuillemin. On circuits and numbers. *IEEE Transactions on Computers*, 43(8):868–879, 1994.

[Vui00] J. Vuillemin. Finite circuits are characterized by 2-algebraic truth tables. In Asian 2000, LNCS 1961, 2000.

[Vui03] J. Vuillemin. Digital algebra and circuits. In LNCS 2772, 2003.

Appendix

Below we sketch the proof of Theorem 2 and illustrate the use of the coinduction proof principle.

Proof of Theorem 2: Let $\mathcal{T}$ be the set of terms built from the function symbols in Σ and from constants c_σ, one for every bitstream σ. The set $\mathcal{T}$ can be turned into a coalgebra $(\mathcal{T}, \langle o, t\rangle : \mathcal{T} \to 2 \times \mathcal{T})$ by defining o and t, first on the constants c_σ as $o(c_\sigma) = \sigma(0)$ and $t(c_\sigma) = c_{\sigma'}$, and then by induction on syntactically more complex terms, following the equations of the theorem. By finality, there exists a unique homomorphism from the coalgebra $\mathcal{T}$ into the coalgebra 2^ω, which assigns to each *syntactic term* $g(c_{\sigma_1}, \ldots, c_{\sigma_r})$ a stream in 2^ω. This stream is then what we define to be the value of $g(\sigma_1, \ldots, \sigma_r)$, where now g denotes a function on bitstreams (of arity r). For details, we refer the reader to [Rut03], where this construction is carried out for (a subset of) the operators of the calculus of streams of real numbers. $\square$

Proof of (2): Let $R = \{\langle \sigma^*, [1] + (\sigma \times \sigma^*)\rangle \mid \sigma \in 2^\omega\} \cup \{\langle \sigma, \sigma\rangle \mid \sigma \in 2^\omega\}$. Then R is a bisimulation relation because $(\sigma^*)(0) = 1 = ([1] + (\sigma \times \sigma^*))(0)$, $(\sigma^*)' = \sigma' \times \sigma^*$, and $([1]+(\sigma\times\sigma^*))' = [0]+(\sigma'\times\sigma^*)+([\sigma(0)]\times(\sigma'\times\sigma^*)) = \sigma'\times\sigma^*$ by idempotency of sum. Now (2) follows by coinduction (Theorem 1). $\square$

Proof of Mixed Laws 5: We only prove $\neg\langle a_0 a_1 \cdots a_n\rangle = \langle \overline{a_0}\,\overline{a_1} \cdots \overline{a_n}\rangle$. The relation $R = \{\langle \neg\langle a_0 a_1 \cdots a_n\rangle, \langle \overline{a_0}\,\overline{a_1} \cdots \overline{a_n}\rangle\,\rangle \mid a_0, \ldots, a_n \in 2\}$ is a bisimulation because $\langle a_0 a_1 \cdots a_n\rangle' = \langle a_1 \cdots a_n a_0\rangle$. The identity follows by coinduction. $\square$

Proof of (5): Substituting intermediate streams for all the connection points in the binary adder circuit, as before, we find that its behaviour is determined by the following two equations (the second of which describes the output of the sum-gate): for all $\sigma, \tau \in 2^\omega$,

$$\begin{aligned} A(\sigma,\tau) &= \sigma \oplus \tau \oplus (X \times \rho) \\ \rho &= ((\sigma \oplus \tau) \wedge (X \times \rho)) + (\sigma \wedge \tau) \end{aligned}$$

Next consider the following two relations $R_0, R_1 \subseteq 2^\omega \times 2^\omega$:

$$\begin{aligned} R_0 = \{&\langle \sigma \mathbin{+\!\!\!+} \tau,\ \sigma \oplus \tau \oplus (X \times \rho)\rangle \mid \\ &\quad \sigma, \tau, \rho \in 2^\omega \text{ s.}t.\ \rho = ((\sigma \oplus \tau) \wedge (X \times \rho)) + (\sigma \wedge \tau)\} \\ R_1 = \{&\langle (\sigma \mathbin{+\!\!\!+} \tau) \mathbin{+\!\!\!+} [1],\ \sigma \oplus \tau \oplus ([1] + (X \times \rho))\rangle \mid \\ &\quad \sigma, \tau, \rho \in 2^\omega \text{ s.}t.\ \rho = ((\sigma \oplus \tau) \wedge ([1] + (X \times \rho))) + (\sigma \wedge \tau)\} \end{aligned}$$

It is straightforward to show that $R_0 \cup R_1$ is a bisimulation relation on 2^ω, from which (5) follows, by coinduction. $\square$

Proof of (6): The set $\{\langle \sigma \mathbin{+\!\!\!+} ([1] \mathbin{+\!\!\!+} \neg\sigma),\ [0]\rangle \mid \sigma \in 2^\omega\}$ is a bisimulation, because $(\sigma \mathbin{+\!\!\!+} ([1] \mathbin{+\!\!\!+} \neg\sigma))(0) = 0$ and $(\sigma \mathbin{+\!\!\!+} ([1] \mathbin{+\!\!\!+} \neg\sigma))' = (\sigma' \mathbin{+\!\!\!+} ([1] \mathbin{+\!\!\!+} \neg\sigma)') \mathbin{+\!\!\!+} [\sigma_0 \cdot (1 \oplus \overline{\sigma_0})] = (\sigma' \mathbin{+\!\!\!+} (([0] \mathbin{+\!\!\!+} \neg(\sigma')) \mathbin{+\!\!\!+} [1 \cdot \overline{\sigma_0}]) \mathbin{+\!\!\!+} [\sigma_0] = \sigma' \mathbin{+\!\!\!+} ([1] \mathbin{+\!\!\!+} \neg(\sigma'))$. This implies (6), by coinduction. $\square$