Equational Axioms of Test Algebra

Marco Hollenberg

CWI, Kruislaan 413, 1098SJ Amsterdam, The Netherlands

1 Introduction

Propositional Dynamic Logic (PDL, [12, 3]) is a widely studied modal program logic set up by simultaneously defining *programs* and *propositions*. A program is viewed as a relation between input states and output states, a purely extensional view on programs. PDL imposes a *regular* structure on the space of programs, allowing sum, composition and iteration of programs. Propositions are statements about states: a proposition either holds at a state or it does not. Accordingly, propositions are equipped with a *boolean* structure. Furthermore, PDL is equipped with operators that turn programs into propositions and vice versa. For instance the program π can be transformed into the proposition that from the current state it is possible to successfully execute π. Similarly a proposition can be turned into a program that returns the input state but succeeds at this only if the proposition is true at this input state: such a program is called a *test*.

Test algebra ([14, 11, 17]) is an attempt to approach PDL algebraically and allow application of common algebraic tools, such as homomorphisms, the Birkhoff theorem, etc. Not only that, common algebraic *questions* also become available. This paper concerns itself with the question of axiomatizing the valid test algebra equalities.

Like PDL, a test algebra is two-sorted: it contains a sort for propositions, with a boolean structure, and one for programs, with a regular algebra, or Kleene algebra ([5, 2]), structure. So the valid equalities also come in two sorts: we have equations linking programs and ones linking propositions. The valid proposition-equations are given by algebraic translations of the Segerberg axioms for PDL ([16, 17]). The valid program-equations may also be axiomatized, but alas the only axiomatization known so far uses not only equations, but also a Π^0_2-statement, known as *separability* ([17]). Such an axiomatization is hardly satisfactory from an algebraic point of view.

In this paper we investigate the possibility of an equational axiomatization for the program equations of test algebra. A finite equational axiomatization is out of the question: the non-finite axiomatizability of Kleene algebra ([15, 2]) carries over to test algebra. However, *relative* to Kleene algebra, test algebra is finitely axiomatizable. That is, adding a finite number of equations to those of Kleene algebra yields a complete axiomatization for test algebra.

As the equations of Kleene algebra are axiomatizable by means of a finite number of equations and quasi-equations (Horn-clauses of equations, see [7]), our result has as an immediate consequence that test algebra is also axiomatizable by such means. Thus: the results of this paper allows one to reason about PDL-programs in an almost purely equational manner.

The paper is layed out as follows. In the next section we introduce relational test algebras and our proposed axiom system TC (Test Calculus). Our proof of completeness (section 4) consists of a reduction to *Kleene algebras with tests* ([9]), which we introduce in section 3. In section 5 we prove that our completeness result is in a sense the best possible: a finite equational axiomatization of test algebras is out of the question. The section also contains a nonfinite axiomatizability result for Kleene algebras with tests, which [9] lacks. Finally, in section 6 we discuss a finite axiomatization for test algebra, involving quasi-equations, using Kozen's axiomatization for Kleene algebra ([7]).

2 Test Algebras

Just as classical propositional logic has an algebraic counterpart in boolean algebras, PDL has a counterpart in test algebras. An algebraist, when first encountering PDL, would recognize a two-sorted algebra, with one sort in which PDL-propositions may be interpreted and another in which PDL-programs are interpreted. The sort for propositions we will refer to as the *boolean sort*, while the other will be called the *program sort.*

The boolean sort has a boolean structure, that is: the structure of a boolean algebra. Boolean algebras are defined as algebras of signature $\{\bot, \neg, \vee\}$ satisfying certain equations. Particular algebras that satisfy these equations are the *set boolean algebras*, algebras of the form:

$$\mathsf{SBA}(S) := (\mathcal{P}(S), \emptyset, {}^{-}, \cup)$$

where S is any set and for any $X \subseteq S$: $\overline{X} = S \setminus X$. Let SBA denote the class of all set boolean algebras. It is well known that the valid equations of SBA are axiomatizable by means of a finite number of equations. Let BA be this set of equations.

The program sort of PDL is equipped with a regular, or Kleene algebra structure ([5, 2]). A *relational Kleene algebra* is determined completely by a set S as follows:

$$\mathsf{RKA}(S) := (\mathcal{P}(S \times S), 0, 1, +, ;, {}^{*})$$

where:

- 0 is the empty relation $\emptyset$. In terms of programs, 0 is the program that always fails to execute.
- 1 is $\mathsf{id}_S := \{(s,s) \mid s \in S\}$, the identity on S. Thus 1, as a program, immediately terminates succesfully, without changing the input state.
- ; is relational composition:

$$R_1; R_2 := \{(s,u) \mid \exists u.(s,u) \in R_1 \;\&\; (u,t) \in R_2\}.$$

A term $t_1; t_2$ will often be denoted simply as $t_1 t_2$.
Composition of programs means executing one after the other succesfully terminates.

- $+$ denotes union of relations. In terms of programs, it is nondeterministic choice.
- Kleene star * denotes reflexive transitive closure: $R^* = \bigcup_{n\geq 0} R^n$, where $R^0 = \mathsf{id}_S$ and $R^{n+1} = R;R^n$. Thus * is iteration: performing a program a finite number of times.

Let RKA denote the class of relational Kleene algebras. In contrast to boolean algebra, the valid relational Kleene algebra equations are not axiomatizable by means of a finite number of equations ([15, 2]). Nevertheless, we are interested in the equational theory of RKA which we denote by KA.

Test algebras, the algebraic variants of PDL, combine these two kinds of algebras into two-sorted algebras with two operators that interact between these sorts. Given an arbitrary (possibly empty) set S, $\mathsf{RTA}(S)$, the *relational test algebra over S*, is defined as a two-sorted algebra $(\mathcal{K}, \mathcal{B}, \Diamond, ?)$ where:

1. $\mathcal{K} = \mathsf{RKA}(S)$ is the relational Kleene algebra over S.
2. $\mathcal{B} = \mathsf{SBA}(S)$ is the set boolean algebra over S.
3. $\Diamond : (\mathcal{K} \times \mathcal{B}) \to \mathcal{B}$ (what Pratt ([14]) calls the *enables* operator) is an operator that takes an element of the program sort (a binary relation on S) and an element of the boolean sort (a subset of S) and produces another boolean element. We will write $\langle R\rangle X$ instead of $\Diamond(R, X)$. It is defined as:
$$\langle R\rangle X := \{s \in S \mid \exists t \in X.sRt\}$$
If the state s is in $\langle R\rangle X$, we say that R *enables* X (in that state). Another way of stating this is that there is at least one execution of R that gives rise to the truth of the *proposition* X.
4. $? : \mathcal{B} \to \mathcal{K}$ (the *test* operator) is a function from the boolean sort to the program sort. It is defined as:
$$X? := \{(s, s) \mid s \in X\}$$
where ? is written in postfix notation.
If X is viewed as a proposition, $X?$ is a program that fails if X is not true but succesfully terminates if it is.

The class of all algebras of the form $\mathsf{RTA}(S)$ will be denoted by RTA. Such algebras will be referred to as *relational test algebras.* Subalgebras of relational test algebras are referred to as *Kripke test structures* in [17] and as *dynamic test set algebras* in [11].

The terms of test algebra coincide precisely with the propositions and programs of PDL. Accordingly, we will sometimes refer to test algebra terms of the boolean sort as PDL-propositions. We use Greek letters $\phi, \psi, \ldots$ and subscripted variants thereof for such terms. Likewise, terms of the program sort will be referred to as PDL-programs. We use $\pi, \pi_0, \pi_1, \ldots$ to denote such terms. Accordingly, the set of program sort variables $A = \{a, b, \ldots\}$ is referred to as the set of *atomic actions* and the set of boolean sort variables $P = \{p, q, \ldots\}$ as the set of *atomic propositions.*

Models for PDL are labeled transition systems (LTSs): models equipped with a binary relation $\xrightarrow{a}$ for every $a \in A$ and unary predicates $p \in P$. Any PDL-program is interpreted as a binary relation on the domain of the LTS. We write $s \xrightarrow{\pi} t$ iff (s,t) is in the interpretation of π. For PDL-propositions ϕ we write $s \Vdash \phi$ if s is in the interpretation of ϕ.

If a program term π is satisfiable (that is, not interpreted by the empty set) in some relational test algebra $\mathsf{RTA}(S)$, under an assignment σ, then from this we can construct an LTS with a π-transition between two states s and t. To be precise, the required LTS has S as its domain, interprets binary relation symbols $\xrightarrow{a}$ by $\sigma(a)$ and unary predicates p by $\sigma(p)$. As $\sigma(\pi) \neq \emptyset$, two states between which there is a π-transition exists in the LTS.

By *unraveling* this LTS, we can even make sure that the term is satisfied in a rooted intransitive tree with all transitions disjoint. Such LTSs will be called *unraveled.* Unraveling works as follows: If $\mathcal{M}$ is an LTS with domain S, then the unraveling of $\mathcal{M}$, $\mathcal{M}^{\text{unr}}$ has as its domain all strings $s_0a_1s_1\ldots a_ns_n$ such that $s_i \xrightarrow{a_i+1} s_{i+1}$ for all $0 \leq i < n$. The only transitions of $\mathcal{M}^{\text{unr}}$ are those of the form:

$$s_0a_1s_1\ldots a_ns_n \xrightarrow{a_{n+1}} s_0a_1s_1\ldots a_ns_na_{n+1}s_{n+1}.$$

Finally:

$$s_0a_1s_1\ldots a_ns_n \Vdash p \text{ iff } s_n \Vdash p.$$

Relational test algebras may be axiomatized by means of the equations of BA, in addition to the following (see [17]). Here a and b are Kleenean variables and p and q are boolean variables.

T1	$\langle a\rangle\bot = \bot$	(normality)
T2	$\langle a\rangle(p \vee q) = \langle a\rangle p \vee \langle a\rangle q$	(additivity)
T3	$\langle 0\rangle p = \bot$	(zero)
T4	$\langle 1\rangle p = p$	(identity)
T5	$\langle a+b\rangle p = \langle a\rangle p \vee \langle b\rangle p$	(plus)
T6	$\langle ab\rangle p = \langle a\rangle\langle b\rangle p$	(composition)
T7	$\langle a^*\rangle p = p \vee \langle a\rangle\langle a^*\rangle p$	(iteration)
T8	$\langle a^*\rangle p = p \vee \langle a^*\rangle(\neg p \wedge \langle a\rangle p)$	(induction)
T9	$\langle p?\rangle q = p \wedge q$	(test diamond)
Sep	$\forall a, b(a \neq b \rightarrow \exists p.\langle a\rangle p \neq \langle b\rangle p)$	(separability)

Thus, for all test algebra terms (of either sort) t_1 and t_2:

$$\mathsf{RTA} \models t_1 = t_2 \quad \text{iff} \quad \mathsf{BA} \cup \{\text{T1}, \ldots, \text{T9}, Sep\} \vdash t_1 = t_2$$

The equations T1 to T9 are just the algebraic translations of the Segerberg axioms ([16]) for PDL, which together with the classical propositional laws are known to be complete. These equations impose no structure whatsoever on the program sort. This problem is addressed by the *separability*-formula.

In this paper we concern ourselves with an equational axiomatization of test algebra. The above does not suffice, solely due to Sep, a Π^0_2-formula.

Sadly, no finite set of equations is complete (see section 5). However, the problem can be shown to lie purely in the Kleene algebra component of test algebra: if we add a certain finite number of equations to those of Kleene algebra, we obtain a complete axiomatization of test algebra. We say that test algebra is finitely axiomatizable *relative to* Kleene algebra.

The equational calculus we consider consists of the following axioms[1]

1. The valid Kleene algebra equations KA.
2. The valid Boolean algebra equations BA.
3. The equations T1, ..., T9.
4. Additional program sort equations:

K1 $\bot ? = 0$	(bottom test)
K2 $(p \vee q)? = p? + q?$	(test sum)
K3 $(p \wedge q)? = p?q?$	(test composition)
K4 $(\langle a \rangle \top)?a = a$	(domain test)

Let TC (*Test Calculus*) denote this set of equations. By the results in [17] it is immediately clear that TC is complete as far as boolean sort equations go. What remains to be proved is that this is also the case for program sort equations.

3 Kleene Algebra with Tests

What sets PDL-programs apart from pure Kleene algebra terms is the presence of *tests* of PDL-propositions. In [9], the equational theory of Kleene algebra extended with tests of purely boolean statements is investigated. These algebras are dubbed *Kleene algebras with tests* (KAT, see also [1, 8]). Although the tests of KAT-algebras are less powerful than those of test algebra, they will be very useful for our purposes: our completeness theorem consists of a reduction of valid TA-equations to valid KAT-equations, for which a completeness theorem is available.

KAT-algebras are two-sorted algebras of the form: $(K, B, 0, 1, +, ;, ^*, ^-)$, where:

- $B \subseteq K$;
- $(K, 0, 1, +, ;, ^*)$ is a Kleene algebra satisfying KA;
- $(\mathcal{B}, 0, 1, ^-, +, ;)$ is a boolean algebra with 0 as bottom, 1 as top, + as join, ; as meet and $^-$ as complement.

Thus, like test algebra, KAT-algebras are two-sorted, but in contrast to test algebras, the boolean sort is a subsort of the program sort. The Kleenean operations $0, 1, +, ;$ are overloaded in the sense that they serve both as boolean operations and as regular operations. * is only defined on the Kleenean sort, while $^-$ is only defined on the boolean sort.

[1] We use some common abbreviations here: $\phi \wedge \psi := \neg(\neg\phi \vee \neg\psi)$, $\top := \neg\bot$ and one we will use in what follows: $[\pi]\phi := \langle\pi\rangle\phi$.

So KAT-algebras are given equationally, by the infinite set KA of Kleene algebra equations and a finite set BA of boolean algebra equations, presented in the language $\{0, 1, ;, +, ^-\}$. Call the union of these two theories KAT.

Relational KAT-algebras are defined as follows. Given any set S, the relational KAT-algebra over S is:

$$\mathsf{RKAT}(S) := (\mathcal{P}(S \times S), \mathcal{P}(\mathsf{id}_S), \emptyset, \mathsf{id}_S, \cup, ;, ^*, ^-)$$

where $\overline{R}$ yields the complement of R relative to id_S for every $R \subseteq \mathsf{id}_S$. Let RKAT denote the class of all relational KAT-algebras.

Kozen and Smith ([9]) also provide a language theoretic model for Kleene algebra with tests, in the spirit of the regular language model for Kleene algebras. We repeat the definition here.

Fix a finite set A of basic Kleenean terms (atomic actions) and a finite set P of basic boolean terms (atomic propositions). Enumerate P as $p_1, \ldots, p_n$ in some fixed, yet arbitrary, order. A *guard* is a term $\gamma_1 \ldots \gamma_n$, where for each $i \in [1, n]$, $\gamma_i \in \{p_i, \overline{p_i}\}$. Let $\mathbb{G}$ denote the finite set of all guards.

Guards correspond in an obvious manner to P-valuations of propositional logic, because guards give a complete specification of which atomic propositions among P are taken to hold. So if α is a guard and t is any boolean sort KAT-term (in other words, t corresponds to a formula of propositonal logic), all of whose variables are among P, we can decide if t 'holds' under the valuation α. If this is the case, we write $\alpha \leq t$.

A *guarded string* is a term of the form:

$$\alpha_0 a_1 \alpha_1 \ldots a_k \alpha_n$$

where each α_i is a guard and each $a_i \in A$. Let $\mathbb{GS}$ be the set of all guarded strings. Note that $\mathbb{G} \subseteq \mathbb{GS}$: guards are also guarded strings, of zero length. Sometimes we will use $\alpha x \beta$ to denote a guarded string, where α and β are its outer guards. This does not mean that $\alpha x \beta$ cannot be a guard: α and β may collapse. Similarly, we will use $x\alpha$ and αx to denote guards, emphasizing the right and left guard respectively.

On sets of guarded strings we may define the operation of *coalesced product*, whereby we concatenate only those guarded strings whose outer guards match. Thus, if X and Y are sets of guarded strings, then:

$$X \diamond Y := \{x\alpha y \mid x\alpha \in X, \alpha y \in Y\}.$$

Using this product, we can define the Kleene star of a set of guarded strings. First define X^n for any $n \in \mathbb{N}$:

$$\begin{aligned} X^0 &:= \mathbb{G} \\ X^{n+1} &:= X \diamond X^n \end{aligned}$$

Then $X^* := \bigcup_{n \in \mathbb{N}} X^n$.

Thus the *guarded string algebra* relative to the sets A and P may be defined as:

$$(\mathcal{P}(\mathbb{GS}), \mathcal{P}(\mathbb{G}), \emptyset, \mathbb{G}, \cup, \diamond, ^*, ^-)$$

where $^-$ is interpreted as complement relative to $\mathbb{G}$.

The *standard interpretation* G assigns a set of guarded strings to any KAT-term all of whose variables occur among $A \cup P$. It is defined as follows:

$$\begin{array}{ll} G(a) & := \{\alpha a \beta \mid \alpha, \beta \in \mathbb{G}\} \\ G(p) & := \{\alpha \mid \alpha \leq p\} \\ G(0) & := \emptyset \\ G(1) & := \mathbb{G} \\ G(t_1 t_2) & := G(t_1) \diamond G(t_2) \\ G(t_1 + t_2) & := G(t_1) \cup G(t_2) \\ G(t^*) & := G(t)^* \\ G(\bar{t}) & := \mathbb{G} \setminus G(t) \end{array}$$

Note that if t is a boolean term, then $G(t)$ is a set of guards. Thus G is a sort-preserving homomorphism from the term-algebra (restricted to terms built from variables out of $A \cup P$) to the guarded string algebra relative to A and P.

Kozen and Smith ([9]) prove the following:

Theorem 1. *For KAT-terms t_1 and t_2 whose variables occur among $A \cup P$:*

$$G(t_1) = G(t_2) \quad \textit{iff} \quad \mathsf{KAT} \vdash t_1 = t_2 \quad \textit{iff} \quad \mathsf{RKAT} \models t_1 = t_2$$

This is not exactly what [9] prove, as they use a different definition of Kleene algebra, involving a finite number of equations and two quasi-equations. Nevertheless, inspection of their completeness-proof yields the above statement.

4 Completeness

In this section we prove completeness of TC, with respect to equations of the program sort. The proof works via a language theoretic model $\mathcal{C}$ of test algebra, which is built up of guarded strings, just like the guarded string algebra of the previous section. In addition, a standard interpretation C is given on $\mathcal{C}$. Thus, if a test algebra equation $\pi_1 = \pi_2$ is valid, C will assign the same set of guarded strings to these terms. If we use TC-equations to put the terms in the equation in the right form, we can even ensure that the interpretation agrees with the standard interpretation G on these terms, modulo an obvious translation, where tests are interpreted as boolean terms by G. By the completeness theorem of [9] this entails that the equation is KAT-derivable, from which we can deduce TC-derivability.

4.1 A Translation from KAT into TA

Let Γ be any finite set of PDL-propositions that is *Fischer-Ladner closed* ([3], see also [4]). That is: Γ is closed under subpropositions and the following rules:

$$\begin{array}{ll} \langle \pi_1 \pi_2 \rangle \phi \in \Gamma & \text{implies } \langle \pi_1 \rangle \langle \pi_2 \rangle \phi \in \Gamma \\ \langle \pi_1 + \pi_2 \rangle \phi \in \Gamma & \text{implies } \langle \pi_1 \rangle \phi, \langle \pi_2 \rangle \phi \in \Gamma \\ \langle \pi^* \rangle \phi \in \Gamma & \text{implies } \langle \pi \rangle \langle \pi^* \rangle \phi \in \Gamma \end{array}$$

Any finite set of PDL-propositions is contained in another one, its *Fischer-Ladner closure*, that is also finite and furthermore Fischer-Ladner closed. Let $\Gamma?$ be $\{\phi? \mid \phi \in \Gamma\}$.

Let P be the set of atomic propositions in Γ. Choose an arbitrary finite set A of atomic actions. Set $\Sigma := A \cup \Gamma?$.

From these ingredients we can build a variety of terms. A *Σ-term* is any term built from elements of Σ and the Kleene algebra operators $0, 1, +, ;, ^*$. In other words, a Σ-term is a PDL-program whose only atomic actions occur in A and whose only tests are of the form $\phi?$ with $\phi \in \Gamma$.

A *Σ-KAT term* is any term built up from Σ and the KAT-operators $0, 1, +, ;,$ $^*, ^-$, where the elements of $\Gamma?$ are treated as undivisibles of the boolean sort. Thus if $\phi?$ and $\psi?$ are elements of $\Gamma?$, then $\overline{\phi?}$ and $\phi?\psi?$ are Σ-KAT terms, but so is $\overline{\phi?\psi?}$.

To any Σ-KAT term we can associate a TA-term $\mathsf{ta}(t)$. The function ta is defined by induction on terms and distributes over all variables and all Kleene algebra operators and on complemented terms it is defined as:

$$\mathsf{ta}(\overline{t}) := ([\mathsf{ta}(t)]\bot)?.$$

Lemma 2. *Let $t_1 = t_2$ be an equation between two Σ-KAT terms. If* $\mathsf{KAT} \vdash t_1 = t_2$ *then* $\mathsf{TC} \vdash \mathsf{ta}(t_1) = \mathsf{ta}(t_2)$.

Proof. It suffices to show that $\mathsf{TC} \vdash \mathsf{ta}(t_1) = \mathsf{ta}(t_2)$ if $t_1 = t_2$ is an instantiation of a KAT-axiom. For the axioms in KA this is obvious, as $\mathsf{KA} \subseteq \mathsf{TC}$ and ta distributes over the Kleene algebra operators.

There are only seven boolean axioms to examine:

$$\begin{aligned}
\overline{p+q} &= \overline{p}\,\overline{q} \\
\overline{pq} &= \overline{p} + \overline{q} \\
\overline{\overline{p}} &= p \\
p + \overline{p} &= 1 \\
p\overline{p} &= 0 \\
pp &= p \\
pq &= qp
\end{aligned}$$

where p and q are boolean variables.

Note that for any boolean sort Σ-KAT term t, $\mathsf{ta}(t)$ is a combination of tests and the operators $0, 1, +, ;$. Any such term is TC-equivalent to a test. For by K1, 0 is equivalent to $\bot?$, by K2 and K3 the class of tests is closed, modulo TC-equivalence, under $+$ and ; and 1 is equivalent to $\top?$, by the following reasoning:

$$\begin{aligned}
\top? &= \top?1 && (\mathsf{KA}) \\
&= \top?(\langle 1\rangle\top)?1 && (\mathsf{K4}) \\
&= (\top \wedge \langle 1\rangle\top)?1 && (\mathsf{K3}) \\
&= (\langle 1\rangle\top)?1 && (\mathsf{BA}) \\
&= 1 && (\mathsf{K4})
\end{aligned}$$

Furthermore, as $\neg\phi = [\phi?]\bot$ is valid, $\mathsf{TC} \vdash (\neg\phi)? = ([\phi?]\bot)?$, so if $\mathsf{ta}(t)$ is provably equivalent to $\phi?$, $\mathsf{ta}(\overline{t})$ is provably equivalent to $(\neg\phi)?$.

So, to prove that for all instances $t_1 = t_2$ (where t_1, t_2 are Σ-KAT terms) of the boolean equations above, $\mathsf{TC} \vdash \mathsf{ta}(t_1) = \mathsf{ta}(t_2)$, it suffices to prove the following in TC:

$$\begin{aligned}
([\phi? + \psi?]\bot)? &= (\neg\phi)?(\neg\psi)? \\
([\phi?\psi?]\bot)? &= (\neg\phi)? + (\neg\psi)? \\
([(\neg\phi)?]\bot)? &= \phi? \\
\phi? + (\neg\phi)? &= 1 \\
\phi?(\neg\phi)? &= 0 \\
\phi?\phi? &= \phi? \\
\phi?\psi? &= \psi?\phi?
\end{aligned}$$

This is left to the reader as an easy exercise.

4.2 Consistent Guarded Strings

Let $\mathcal{G}$ be the guarded string algebra with respect to A and $\Gamma?$, where $\Gamma?$ is enumerated in some fixed, yet arbitrary, way as $\phi_1?, \ldots, \phi_n?$. The standard interpretation G assigns a set of guarded strings in $\mathcal{G}$ to any Σ-KAT term.

A guarded string x is called *consistent* if the TA-term $\mathsf{ta}(x)$ associated with it is *satisfiable*, that is: if $\mathsf{RTA} \not\models \mathsf{ta}(x) = 0$.

Let $\mathbb{C}$ denote the set of consistent guards, and $\mathbb{CS}$ the set of consistent guarded strings. We build a test algebra $\mathcal{C}$ out of these ingredients:

- Its program domain is $\mathcal{P}(\mathbb{CS})$. The Kleene algebra operations are defined as in the guarded string algebra, except that 1 is not interpreted by $\mathbb{G}$, which may contain inconsistent guards, but by $\mathbb{C}$.
- Its boolean domain is $\mathcal{P}(\mathbb{C})$. The operations $\bot, \neg$ and $\vee$ are interpreted as $\emptyset$, complement relative to $\mathbb{C}$, and union respectively.
- The enables operator is defined as:
$$\langle R \rangle X := \{\alpha \in \mathbb{C} \mid \exists \alpha x \beta \in R . \beta \in X\}$$
- The test operator is defined as: $X? := X$. This is possible precisely because $\mathbb{C} \subseteq \mathbb{CS}$.

To show that $\mathcal{C}$ is really an algebra, we have to prove that its domains are closed under the operations. This is trivial for all but coalesced product. So we need to prove that if $x\alpha$ and αy are two consistent guarded strings, then $x\alpha y$ is also a consistent guarded string. In matters of consistency it is easier to work with LTSs instead of with relational test algebras with assignments, even though these are equivalent. To show closure under coalesced product, we need the notion of *Γ-bisimulation*.

Definition 3. Two points s and t in a labeled transition system (LTS) are *Γ-equivalent* iff they agree on all formulas in Γ. Two points s and t are *Γ-bisimilar* if:

Invariance: They agree on all atomic propositions (by assumption these occur in Γ).

Zig: If $s \xrightarrow{a} s'$ for some atomic action a then there is some a-successor t' of t that is Γ-equivalent to s'.
Zag: Vice versa: if $t \xrightarrow{a} t'$ for some Γ-variable a then there is an s' that is Γ-equivalent to t' such that $s \xrightarrow{a} s'$.

Calling this notion *bisimulation* may be misleading, as the zig- and zag-clauses do not require Γ-bisimilarity between the successors, but only Γ-equivalence. A better name would have been *one-step Γ-bisimilarity*, but for the sake of brevity we stick to 'Γ-bisimilarity'.

Lemma 4. *Let $\mathcal{M}$ and $\mathcal{N}$ be two unraveled LTSs and let $s \in \mathcal{M}$, $t \in \mathcal{N}$. If s and t are Γ-bisimilar then they are Γ-equivalent.*

Proof. We proceed by induction on the complexity of formulas in Γ. The base cases for $\perp$ and the atomic propositions p are trivial. Likewise for the booleans.
Let π be a program such that for all tests $\psi?$ in π, we have already proved that $s \Vdash \psi$ iff $t \Vdash \psi$.

Claim 1: Suppose $s \xrightarrow{\pi} s$. Then $t \xrightarrow{\pi} t$ also holds. This may be proved by induction on the size of π, using the fact that $\mathcal{M}$ and $\mathcal{N}$ are unraveled.

Claim 2: For any ϕ such that $\langle\pi\rangle\phi \in \Gamma$, if $s \xrightarrow{\pi} s' \Vdash \phi$ with $s \neq s'$ then $t \Vdash \langle\pi\rangle\phi$. The proof again proceeds by induction on π, using Γ-bisimilarity of s and t for the base case where π is an atomic action, and Claim 1 for the ;-case.

We continue with the proof of the lemma. Suppose $s \Vdash \langle\pi\rangle\phi$ for some $\langle\pi\rangle\phi \in \Gamma$. As induction hypothesis assume that s and t agree on all proper subpropositions of $\langle\pi\rangle\phi$. In particular they agree on all tests that occur in π and on ϕ. If $s \xrightarrow{\pi} s \Vdash \phi$ we conclude that $t \xrightarrow{\pi} t$ (Claim 1) and that $t \Vdash \phi$ (induction hypothesis on ϕ). If $s \xrightarrow{\pi} s' \Vdash \phi$ with $s \neq s'$ we can use Claim 2: $t \Vdash \langle\pi\rangle\phi$.

Lemma 5. *If $\alpha a\beta$ and βx are two consistent guarded strings then $\alpha a\beta x$ is also a consistent guarded string.*

Proof. We need to prove that $\alpha a\beta x$ is consistent. By assumption $\alpha a\beta$ and βx are consistent. Let $\mathcal{M}$ be an unraveled LTS rooted in s such that $\mathsf{ta}(\alpha)$ succeeds on s and there is an a-step from s to s' where $\mathsf{ta}(\beta)$ succeeds. Let $\mathcal{N}$ be an unraveled LTS such that there is a $\mathsf{ta}(\beta x)$-transition emanating from the root of $\mathcal{N}$. Now replace the submodel generated by s' in $\mathcal{M}$ by $\mathcal{N}$. The resulting model $\mathcal{M}'$ has a root that is Γ-bisimilar to that of $\mathcal{M}$. By lemma 4 this implies Γ-equivalence, hence $\mathsf{ta}(\alpha)$ succeeds at the root of $\mathcal{M}'$. Thus there is an $\mathsf{ta}(\alpha)a\mathsf{ta}(\beta)$-step from the root of $\mathcal{M}'$ to the root of $\mathcal{N}$, from which we may continue with an $\mathsf{ta}(x)$-transition. So $\mathsf{ta}(\alpha a\beta x)$ is satisfiable in $\mathcal{M}'$.

Corollary 6. *Consistent guarded strings are closed under* coalesced product*: if $x\alpha$ and αy are two consistent guarded strings then $x\alpha y$ is also a consistent guarded string.*

Theorem 7. *$\mathcal{C}$ is a subalgebra of some relational test algebra $\mathcal{A}$. Hence $\mathcal{C} \models$ TC.*

Proof. Let $\mathcal{A}$ be the relational test algebra over $\mathbb{CS}$, considered as a set. That is: $\mathcal{A} = \mathsf{RTA}(\mathbb{CS})$. We will provide an embedding of $\mathcal{C}$ into $\mathcal{A}$. Because test algebras are two-sorted, this embedding is given by a function k from $\mathcal{P}(\mathbb{CS})$ to binary relations over $\mathbb{CS}$:

$$k(R) := \{(x\alpha\, ,\, x\alpha y) \in \mathbb{CS}^2 \mid \alpha \in \mathbb{C},\ \alpha y \in R\}$$

and a function b from $\mathcal{P}(\mathbb{C})$ to $\mathcal{P}(\mathbb{CS})$:

$$b(X) := \{x\alpha \in \mathbb{CS} \mid \alpha \in X\}.$$

The proof that this is really an embedding is left to the reader.

We provide $\mathcal{C}$ with a standard interpretation C. On the variables C is defined as follows:

$$C(a) := \{\alpha a \beta \in \mathbb{CS} \mid \alpha, \beta \in \mathbb{C}\}$$
$$C(p) := \{\alpha \in \mathbb{C} \mid \alpha \leq p?\}$$

This interpretation can be homomorphically extended to all TA-terms.

Lemma 8. *For $\phi \in \Gamma$: $C(\phi) = \{\alpha \in \mathbb{C} \mid \alpha \leq \phi?\}$.*

Proof. By induction on ϕ. The modal case $\langle\pi\rangle\phi$ is of course where the difficulties are to be found. Here a lemma is useful, stating that $\sum \mathsf{ta}[C(\pi)] = \pi$ is valid in all test algebras assuming the lemma has been proved for all tests $\psi?$ in π. Here the sum may be infinite. It finds a natural interpretation in infinite union.

Corollary 9. *If α is a consistent guard then $C(\mathsf{ta}(\alpha)) = \{\alpha\}$.*

4.3 A Translation from TA into KAT

Now we have two algebras. On the one hand we have the guarded string algebra $\mathcal{G}$ relative to A and $\Gamma?$. On the other we have the algebra $\mathcal{C}$ which is built up from a subset of the guarded strings that $\mathcal{G}$ is made out of, namely the consistent ones.

Theorem 10. *For any Σ-term π there is a Σ-KAT term $\widehat{\pi}$ such that:*

1. $\mathsf{TC} \vdash \pi = \mathsf{ta}(\widehat{\pi})$.
2. $C(\pi) = G(\widehat{\pi})$.

Proof. We define $\widehat{t}$ by induction on t.

- $\widehat{1} := \sum \mathbb{C}$.

  First we prove that for any guard $\alpha \in \mathbb{C}$: $\mathsf{TC} \vdash \mathsf{ta}(\alpha) = (\langle \mathsf{ta}(\alpha) \rangle \top)?$. $\mathsf{ta}(\alpha)$ is of the form $\psi_1? \ldots \psi_n?$. We reason as follows in TC:

$$\begin{aligned} \psi_1? \ldots \psi_n? &= (\psi_1 \wedge \ldots \wedge \psi_n)? && \text{(K3)} \\ &= (\psi_1 \wedge \ldots \wedge \psi_n \wedge \top)? && \mathsf{BA} \\ &= (\langle (\psi_1 \wedge \ldots \wedge \psi_n)? \rangle \top)? && \text{(T9)} \\ &= (\langle \psi_1? \wedge \ldots \wedge \psi_n? \rangle \top)? && \text{(K3)} \end{aligned}$$

  Next we note that $\mathsf{RTA} \models \top = \bigvee_{\alpha \in \mathbb{C}} \langle \mathsf{ta}(\alpha) \rangle \top$, as $\mathbb{C}$ contains all *consistent* guards. By completeness of TC for test algebra equations of the boolean sort, this equation is derivable. Thus we can reason as follows:

$$\begin{aligned} \mathsf{ta}(\textstyle\sum \mathbb{C}) &= \textstyle\sum \mathsf{ta}[\mathbb{C}] \\ &= \textstyle\sum_{\alpha \in \mathbb{C}} (\langle \mathsf{ta}(\alpha) \rangle \top)? \\ &= (\textstyle\bigvee_{\alpha \in \mathbb{C}} \langle \mathsf{ta}(\alpha) \rangle \top)? && \text{(K2)} \\ &= \top? \\ &= 1 \end{aligned}$$

  The second condition of the theorem is satisfied because for any guard α: $G(\alpha) = \{\alpha\}$. So $C(1) = \mathbb{C} = \bigcup_{\alpha \in \mathbb{C}} G(\alpha) = G(\sum \mathbb{C})$.
- $\widehat{a} = \sum C(a)$.

  Using that $\mathsf{TC} \vdash 1 = \sum \mathsf{ta}[\mathbb{C}]$:

$$\begin{aligned} a &= 1a1 && (\mathsf{KA}) \\ &= (\textstyle\sum \mathsf{ta}[\mathbb{C}]) a (\textstyle\sum \mathsf{ta}[\mathbb{C}]) \\ &= \textstyle\sum_{\alpha, \beta \in \mathbb{C}} \mathsf{ta}(\alpha) a \mathsf{ta}(\beta) && (\mathsf{KA}) \end{aligned}$$

  This sum may contain unsatisfiable programs π. But if $\mathsf{RTA} \models \pi = 0$, then $\mathsf{RTA} \models \langle \pi \rangle \top = \bot$, so by completeness for boolean sort equations: $\mathsf{TC} \vdash \langle \pi \rangle \top = \bot$. π may thus be proved equal to 0 as follows:

$$\begin{aligned} \pi &= \langle \pi \rangle \top? \, \pi && \text{(K4)} \\ &= \bot? \, \pi \\ &= 0\pi && \text{(K1)} \\ &= 0 && (\mathsf{KA}) \end{aligned}$$

  Thus the unsatisfiable terms in the above sum may be proved equal to 0 and so removed from this sum by Kleene algebra reasoning.

  The second condition is also satisfied, as $G(\alpha a \beta) = \{\alpha a \beta\}$.
- $\widehat{\phi?} = \sum C(\phi)$.

  Clearly $\mathsf{RTA} \models \phi = \bigvee_{\alpha \in C(\phi)} \langle \mathsf{ta}(\alpha) \rangle \top$, so this equation must be derivable, by completeness of TC for boolean sort equations. Thus:

$$\begin{aligned} \phi? &= (\textstyle\bigvee_{\alpha \in C(\phi)} \langle \mathsf{ta}(\alpha) \rangle \top)? \\ &= \textstyle\sum_{\alpha \in C(\phi)} (\langle \mathsf{ta}(\alpha) \rangle \top)? && \text{(K2)} \\ &= \textstyle\sum_{\alpha \in C(\phi)} \mathsf{ta}(\phi) \end{aligned}$$

  The second condition is satisfied, by lemma 8

– $\widehat{(\_)}$ distributes over the operators $0, +, ;, ^*$.

Theorem 11 Completeness of TC. *If π_1 and π_2 are two PDL-programs such that* $\mathsf{RTA} \models \pi_1 = \pi_2$ *then* $\mathsf{TC} \vdash \pi_1 = \pi_2$.

Proof. Define A to contain all atomic actions occurring in either π_1 or π_2 and let Γ be the Fisher-Ladner closure of all PDL-propositions occurring in these same programs. From A and Γ? we can build the guarded string model $\mathcal{G}$ and the consistent guarded string model $\mathcal{C}$, as defined above, together with their standard interpretations.

By theorem 7, $\mathcal{C}$ is a subalgebra of some relational test algebra. As equations are preserved under subalgebras, $\mathcal{C} \models \pi_1 = \pi_2$. In particular, $C(\pi_1) = C(\pi_2)$.

By theorem 10, $\mathsf{TC} \vdash \pi_i = \mathsf{ta}(\widehat{\pi_i})$ for $i \in \{1, 2\}$. Also $G(\widehat{\pi_1}) = C(\pi_1) = C(\pi_2) = G(\widehat{\pi_2})$, which implies, by theorem 1, that $\widehat{\pi_1} = \widehat{\pi_2}$ is derivable using the KAT-equations, hence $\mathsf{ta}(\widehat{\pi_1}) = \mathsf{ta}(\widehat{\pi_2})$ is derivable in TC by lemma 2. We conclude $\mathsf{TC} \vdash \pi_1 = \pi_2$.

5 Nonfinite Axiomatizability

We have proved that test calculus is complete for relational test algebra. Test calculus contains an infinite component, namely the set of valid Kleene algebra equations. In this section we show that we cannot do any better if we stick to equations in our axiomatization.

Our problem would be solved immediately if the valid Kleene algebra equations were themselves axiomatizable by means of a finite number of equations. That this is not the case was shown in [15] (for a proof in English, see [2]). There it is shown that for any finite set T of valid Kleene algebra equations there is another valid Kleene algebra equation $\pi_1 = \pi_2$ and an algebra $\mathcal{K}$ such that $\mathcal{K} \models T$, yet $\mathcal{K} \not\models \pi_1 = \pi_2$. There is no need to go into the details of the construction of this algebra $\mathcal{K}$, but for one: in [2] the algebra $\mathcal{K}$ satisfies $\forall xy(xy = 0 \rightarrow (x = 0 \vee y = 0))$. This will be useful in our proof.

First let us prove that Kleene algebra with tests are not finitely equationally axiomatizable. Suppose that they *are* finitely axiomatizable, say by a set of equations T. We may assume that T contains the boolean algebra equations BA, because this is a finite set. As these BA-equations allow us to move complement inwards, we can assume that T is of the form $U \cup \mathsf{BA}$, where U is a set of equations where the complement operator $^-$ is only applied to boolean variables.

Let P be the finite set of boolean variables in U. On P we can define *assignments* $\sigma : P \rightarrow \{0, 1\}$, where 0 and 1 are terms. Such assignments can be extended to terms occurring in U by distributing over the Kleene algebra operators and defining:

$$\sigma(\overline{p}) := \begin{cases} 1 \text{ if } \sigma(p) = 0. \\ 0 \text{ if } \sigma(p) = 1. \end{cases}$$

Then for any term in U, $\sigma(U)$ is a Kleene algebra term. Now let $U_\sigma := \{\sigma(t_1) = \sigma(t_2) \mid t_1 = t_2 \in U\}$. This set only contains valid Kleene algebra equations. The same can be said for

$$U' := \bigcup\{U_\sigma \mid \sigma : P \to \{0,1\}\} \cup \{0a = 0, 1a = a\}.$$

This is still a finite set, so by the nonfinite axiomatizability result for Kleene algebra there must be an algebra $\mathcal{K}$ satisfying U' and a valid Kleene algebra equation $\pi_1 = \pi_2$ that is not satisfied by $\mathcal{K}$. This algebra can be turned into an algebra of the KAT-type by letting the boolean domain B be $\{0,1\}$ and defining $\overline{0} = 1$ and $\overline{1} = 0$, keeping everything else the same. We needed the last two equations of U' to make sure that B is closed under the operations $+$ and ;. Let $\mathcal{K}'$ be this new algebra. Clearly it satisfies BA, as the boolean component is either the trivial algebra (if $0 = 1$, in which case $\mathcal{K}$ is also trivial) or the prototypical two-element boolean algebra (here we again need the last two equations of U'). But it also satisfies U. For suppose $t_1 = t_2 \in U$ and that τ is some $\mathcal{K}$-assignment. Let σ be the restriction of τ to the variables P. Because the boolean domain consists of $\{0,1\}$, σ is a function from P to $\{0,1\}$. Thus $\mathcal{K}' \models t_1 = t_2[\tau]$ iff $\mathcal{K}' \models \sigma(t_1) = \sigma(t_2)[\tau]$. The latter is true by assumption. We conclude that $\mathcal{K}'$ is an algebra satisfying T that does not satisfy the valid Kleene algebra equation $\pi_1 = \pi_2$.

This suffices for the proof that KAT-algebra is not finitely equationally axiomatizable. We note that the algebra $\mathcal{K}'$ was such that $\forall xy(xy = 0 \to (x = 0 \lor y = 0))$ held and that the boolean domain consisted purely of 0 and 1.

Now to prove the same for test algebra. For this we need a refinement of lemma 10. In the TC-proofs of $\pi = \mathsf{ta}(\widehat{\pi})$ not all of test calculus is used, but only a finite part, namely $\mathsf{TC} \setminus \mathsf{KA}$ plus the idempotent semiring axioms IS:

$$\begin{array}{ll}
a(bc) = (ab)c & a + (b + c) = (a + b) + c \\
a1 = 1a = a & a + a = a \quad a + b = b + a \\
a0 = 0a = 0 & a + 0 = a \\
a(b + c) = (ab) + (ac) & (a + b)c = (ac) + (bc)
\end{array}$$

Define $\mathsf{TC}_0 := (\mathsf{TC} \setminus \mathsf{KA}) \cup \mathsf{IS}$.

Lemma 10 can thus be sharpened to: for any Σ-term π there is a Σ-KAT term $\widehat{\pi}$ such that $\mathsf{TC}_0 \models \pi = \mathsf{ta}(\widehat{\pi})$ and $C(\pi) = G(\widehat{\pi})$ (where these are relative to some choice of A and Γ of course). Thus, if $\mathsf{RTA} \models \pi_1 = \pi_2$, there are terms $\widehat{\pi_i}$ such that: $\mathsf{TC}_0 \vdash \pi_i = \mathsf{ta}(\widehat{\pi_i})$ (for $i = 1, 2$) and $\mathsf{KAT} \vdash \widehat{\pi_1} = \widehat{\pi_2}$.

Now suppose T is a finite set of sound test algebra equations. Let U consist of all program sort equations in T. We will show that there is a program sort equation that is not implied by T.

For every equation $\pi_1 = \pi_2 \in U$ there is a KAT-derivable equation $\widehat{\pi_1} = \widehat{\pi_2}$ such that $\mathsf{TC}_0 \vdash \pi_i = \mathsf{ta}(\widehat{\pi_i})$, for $i = 1, 2$. If we uniformly replace in this KAT-equation all tests by boolean variables, what remains is still a KAT-derivable equation, that moreover implies (by substitution) the original $\widehat{\pi_1} = \widehat{\pi_2}$. Let U' consist of all KAT-equations we obtain in this manner. U' is thus a finite set.

By the nonfinite axiomatizability proof for KAT, there is an algebra $\mathcal{K} = (K, B, 0, 1, +, ;, ^*, ^-)$ such that $\mathcal{K} \models U' \cup \mathsf{IS}$ yet that does not satisfy all valid

Kleene algebra equations. We may assume that the boolean component $B = \{0,1\}$. Furthermore $xy = 0$ iff $x = 0$ or $y = 0$.

This algebra can be turned into a TA-type algebra:

$$\mathcal{T} := ((K, 0, 1, +, ;, ^*),\ (B, 0, ^-, +),\ \diamond,\ ?)$$

by defining $x? := x$ and:

$$\langle x\rangle y := \begin{cases} 0 \text{ if } x = 0 \text{ or } y = 0; \\ 1 \text{ otherwise.} \end{cases}$$

This algebra satisfies all the equations in TC_0 and furthermore, for every $\pi_1 = \pi_2 \in U$: $\mathcal{T} \models \mathsf{ta}(\widehat{\pi_1}) = \mathsf{ta}(\widehat{\pi_2})$. As an example, consider the equation T6: $\langle ab\rangle p = \langle a\rangle\langle b\rangle p$. There are four cases to consider:

1. $a = 0$. Immediately we have $\langle a\rangle\langle b\rangle p = 0$, but by IS we also have $ab = 0$, so $\langle ab\rangle p = 0$.
2. $b = 0$. Then $\langle b\rangle p = 0$, so $\langle a\rangle\langle b\rangle p = 0$. Again, use a semiring axiom to derive $ab = 0$ and we also get $\langle ab\rangle p = 0$.
3. $p = 0$. Then $\langle b\rangle p = 0$, so $\langle a\rangle\langle b\rangle p = 0$. But $p = 0$ also gives us $\langle ab\rangle p = 0$.
4. None of the above (in particular then: $p = 1$). By our assumption that $xy = 0$ iff $x = 0$ or $y = 0$ we get that $ab \neq 0$, so $\langle ab\rangle p = 1$. But we also have $\langle b\rangle p = 1$, so $\langle a\rangle\langle b\rangle p = 1$.

So we have an algebra $\mathcal{T}$ that satisfies the TC_0 equations, and for every $\pi_1 = \pi_2 \in U$, $\mathcal{T}$ also satisfies $\mathsf{ta}(\widehat{\pi_1}) = \mathsf{ta}(\widehat{\pi_2})$. By the sharpened version of lemma 10, $\mathcal{T} \models U$. Alas, there is a valid Kleene algebra equation that $\mathcal{T}$ does *not* satisfy, so T is not complete.

6 Conclusion

We have axiomatized test algebra equations and achieved a finite equational axiomatization relative to Kleene algebra. The previous section showed that the Kleene algebra component cannot be replaced by a finite set of test algebra equations.

There is thus a tradeoff: on the one hand we have a sound and complete axiomatization using an infinite number of equations (this paper), on the other we have a finite axiomatization that uses a Π^2_0-axiom ([17]). However, if we replace the Kleene algebra component of the test calculus with any theory that is sound in test algebra and implies all the valid Kleene algebra equations, we get another sound and complete theory. One such theory is presented in [7]. This theory consists of a finite number of equations and two quasi-equations:

$$ab \leq b \rightarrow a^*b \leq b \qquad ba \leq b \rightarrow ba^* \leq b$$

where $x \leq y$ abbreviates $x + y = y$ (this theory is also used in [9] to define Kleene algebras). Thus, a finite axiomatization is possible with a purely Π^0_1-theory, in fact only quasi-equations are necessary. We do not need to reprove the completeness theorem: it is a simple matter of combining the theorem of this paper and that of [7].

Acknowledgements
I would very much like to thank Wan Fokkink, Frederick Smith, Hajnal Andréka, Jan Bergstra, Luca Aceto and Albert Visser (in no particular order).

References

1. E. Cohen, D. Kozen and F. Smith. The complexity of Kleene algebra with tests. Technical Report 96-1598, Computer Science Department, Cornell University, July 1996.
2. J.H. Conway. *Regular Algebra and Finite Machines.* Chapman and Hall, London, 1971.
3. M.J. Fisher and R.E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.
4. R. Goldblatt. *Logics of Time and Computation. 2nd edition*, volume 7 of *CSLI Lecture Notes.* CSLI Publications, Stanford, 1992.
5. S.C. Kleene. Representation of events in nerve nets and finite automata. In Shannon and McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.
6. D. Kozen. On induction versus *-continuity. In D. Kozen, editor, *Proceedings Workshop on Logics of Programs 1981*, volume 131 of *LNCS*, pages 167-176, 181.
7. D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110:2:366–390, May 1994.
8. D. Kozen. Kleene algebra with tests. Transactions on Programming Languages and Systems, 427-443, May 1997.
9. D. Kozen and F. Smith. Kleene algebra with tests: completeness and decidability. Proc. 10th Int. Workshop on Computer Science Logic (CSL'96), ed. D. van Dalen and M. Bezem, Utrecht, The Netherlands, Springer-Verlag LNCS volume 1258, 244–259, September 1996.
10. D. Krob. A complete system of B-rational identities. *Theoretical Computer Science*, 89(2):207–343, October 1991.
11. I. Németi. Dynamic algebras of programs. In *Proceedings Fundamentals of Computation Theory*, volume 117 of *LNCS*, pages 281–290. Springer-Verlag, 1981.
12. V. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.
13. V. Pratt. Dynamic algebras as a well-behaved fragment of relation algebras. In D. Pigozzi, editor, *Proceedings Conference on Algebra and Computer Science*, volume 425 of *LNCS*, pages 77–110. Springer, June 1988.
14. V. Pratt. Dynamic algebras: examples, constructions, applications. *Studia Logica*, 50:571–605, 1991.
15. V.N. Redko. On defining relations for the algebra of regular events. *Ukrainskii Matematicheskii Zhurnal*, 16:120–126, 1964. In Russian.
16. K. Segerberg. A completeness theorem in the modal logic of programs. In T. Traczyk, editor, *Universal algebra and applications*, volume 9 of *Banach Centre Publications*, pages 31–46. PWN - Polish Scientific Publishers, Warsaw, 1982.
17. V. Trnková and J. Reiterman. Dynamic algebra with test. *Journal of Computer and System Sciences*, 35:229–242, 1987.