Equational axioms of test algebra

Marco Hollenberg
Utrecht University, Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht, the Netherlands
hollenb@phil.ruu.nl
http://www.phil.ruu.nl/home/marco/

December 9, 1996

Abstract

We present a complete axiomatization of *test algebra* ([24, 18, 29]), the two-sorted algebraic variant of Propositional Dynamic Logic (PDL, [21, 7]). The axiomatization consists of adding a finite number of equations to any axiomatization of Kleene algebra ([15, 26, 17, 4]) and algebraic translations of the Segerberg ([27]) axioms for PDL. Kleene algebras are not finitely axiomatizable ([25, 6]), so our result does not give us a finite axiomatization of test algebra: in fact, no finite equational axiomatization exists. We also present a single-sorted version of test algebra, using the notion of *dynamic negation* ([9, 2, 11]), to which the previous results carry over.

1 Introduction

Propositional Dynamic Logic (PDL, [21, 7]) is a widely studied modal logic, capable of reasoning about labeled transition systems (LTSs), and thus about any objects that use LTSs as their models, such as computational processes, the intended domain of application. The logic is set up by simultaneously defining *programs* and *propositions*. A program is viewed in a purely extensional way, as a relation between input states and output states. 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*. For a general picture of this *dynamic architecture*, with a space of propositions, a space of programs and so-called *modes* (transformations of propositions into programs) and *projections* (the other way around) between them, see [3].

Dynamic algebra ([13, 22, 24]) may be viewed as an attempt to approach PDL algebraically and allow application of common algebraic tools. Not only that, common algebraic *questions* also become available. In particular, the question of axiomatizing the equalities that are valid in dynamic algebras arises. Like PDL, a dynamic 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 ([12, 6]), structure. So the valid equalities also come in two sorts: we have equations linking programs and ones linking propositions.

Dynamic algebra is however lopsided in that it only has projections (in the sense of [3]) but no modes. So dynamic algebra corresponds to a version of PDL that has test-free programs only. This implies that the valid program-equations of dynamic algebra coincide with those of Kleene algebra. The valid proposition-equations are given by algebraic translations of the Segerberg axioms for PDL ([27]). Kleene algebra is not finitely axiomatizable ([25, 6]), hence dynamic algebra is not either.

A true algebraic counterpart to PDL is *test algebra* ([24, 18, 29]). These are simply dynamic algebras that are equipped with the mode of turning a proposition into its test. Now the valid program-equations will strictly include those of Kleene algebra, as program-terms can now include tests. This paper concerns itself with the evident question of axiomatizing the valid test algebra equalities.

The paper is organized as follows. The next section contains the definition of test algebra and the proposed axiomatization. On the way, Kleene algebra, various models thereof, PDL, dynamic algebra and axiomatizations of all of these will also be discussed. The axiomatization of test algebra is arrived at by adding a finite number of equations to a necessarily infinite or rule-based axiomatization of Kleene algebra combined with algebraic PDL-axioms. The section after that contains the main proofs of the paper, with as the principal lemma a proof that using only a finite number of valid equations, any program-equation can be transformed into a normalform that is valid using only Kleene algebra axioms. The method of the proof is a combination of completeness proofs to be found in [16] and [11]. The lemma has as immediate corollaries the completeness of our system and the fact that test algebras cannot be finitely axiomatized: the infinite axiomatization of Kleene algebra contained in the system for test algebra cannot be replaced by a finite equational one that mentions tests in its axioms. Finally, we show that test algebras have single-sorted counterparts to which all the previous results apply. The main idea is adding *dynamic negation* ([9, 2, 11]) to the signature of Kleene algebras, that turns any program into a test whether this program *cannot* succeed with the current state as input. The paper concludes with some suggestions for future research.

2 Algebras and axiomatizations

2.1 Kleene algebras

Kleene algebras are certain algebras of a signature containing $0, 1$ (both constants), $+, ;$ (both binary) and $*$ (unary). We describe two classes of such algebras which generate the same variety (i.e. satisfy the same equations).

First there is the class of *relational Kleene algebras*, which we denote by REL. In such algebras the domain is the set of all binary relations on some set S. 0 is then interpreted as the empty relation, 1 as the identity relation on S:

$$\mathsf{id}_S = \{(s, s) \mid s \in S\}$$

(also known as the *diagonal*), $+$ as union, $;$ as relational composition:

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

and $*$ as reflexive transitive closure: $R^* = \bigcup_{n \geq 0} R^n$, where $R^0 = \mathsf{id}_S$ and $R^{n+1} = R; R^n$.

Another class of Kleene algebras is the class REG of *regular language models*, which is the class of all algebras $\mathbf{Reg}_\Sigma$, defined as follows. Σ is any finite alphabet. The domain of $\mathbf{Reg}_\Sigma$ consists of all sets $S \subseteq \Sigma^*$ (i.e. all sets of finite strings over the alphabet Σ). 0 is interpreted by the empty set, 1 by $\{\epsilon\}$ (where ϵ is the empty string), $+$ by union, $S; T := \{\sigma \cdot \tau \mid \sigma \in S, \tau \in T\}$ (where $\cdot$ denotes string concatenation) and S^* consists of all finite strings $\sigma_1 \cdot \ldots \cdot \sigma_n$ where each σ_i is an element of S.

For any Σ, $\mathbf{Reg}_\Sigma$ can be embedded into the relational Kleene Algebra with as its domain all binary relations over Σ^*. $h : S \mapsto \{(\sigma, \sigma \cdot \tau) \mid \tau \in S\}$ is the desired embedding. This implies that any quasi-equation (i.e. any Horn-clause of equations) that is valid in REL will also be valid in REG.

Kleene ([12]) first studied the equational theory of regular language models. This theory has become known as the *algebra of regular events*. Kleene posed axiomatization as an open problem. Redko ([25]) first proved that a *finite* equational axiomatization is out of the question (see [6] for a proof of this fact in English). Not discouraged by this, however, a number of authors have produced axiomatizations for Kleene algebra. Of these, Salomaa ([26]) was the first, giving *two* rule-based complete axiomatizations. Axiomatizations involving *schematic* equations (representing an infinite number of equations) have also appeared (see [17] and [4]). We focus in this paper on the relatively

transparent axiomatization of Kozen ([15]): it involves a finite number of equations and two quasi-equations (Horn-clauses of equations). These quasi-equations may be viewed as *rules*, which relate Kozen's axiomatization to those of Salomaa. However, Salomaa's rules involve complex side-conditions (of the sort: 'π does not contain the empty word') whereas Kozen's rules have *no* side-conditions.

For reference, we present Kozen's system. Define KC (for 'Kleene Calculus' or 'Kozen's Calculus') to be the set of the following quasi-equations:

K1	$a;(b;c) = (a;b);c$	(associativity of ;)
K2	$a;1 = a$	(identity left)
K3	$1;a = a$	(identity right)
K4	$0;a = 0$	(zero left)
K5	$a;0 = 0$	(zero right)
K6	$a+(b+c) = (a+b)+c$	(associativity of +)
K7	$a+b = b+a$	(commutativity)
K8	$a+a = a$	(idempotency)
K9	$(a+b);c = (a;c)+(b;c)$	(distributivity left)
K10	$a;(b+c) = (a;b)+(a;c)$	(distributivity right)
K11	$a+0 = a$	(zero addition)
K12	$a^* = 1+(a;a^*)$	($*$-expansion)
K13	$a;b \leq b \;\Rightarrow\; a^*;b \leq b$	(induction left)
K14	$b;a \leq b \;\Rightarrow\; b;a^* \leq b$	(induction right)

where $t \leq u$ is defined as $t+u = u$. In both the relational interpretation and in language models, $\leq$ is interpreted by subset inclusion. Note that all the operations are provably monotonous with respect to $\leq$.

An equation is derivable from KC iff its universal closure is first-order derivable from the universal closure of the KC-axioms (where the $\Rightarrow$ in the two proper quasi-equations K13 and K14 is interpreted as classical implication). This is equivalent to saying that an equation is derivable from KC iff it is derivable from axioms K1-12 using the equational logic rules expressing that $=$ is an equivalence relation, uniform substitution and K13 and K14, now interpreted as rules: in this interpretation K13 says that if $t;u \leq u$ (where t and u are any terms) is derivable then we may derive $t^*;u \leq u$ and similarly for K14.

Remark: Kozen's system in [15] has an extra equation, $1+(a^*;a) \leq a^*$, but this is already derivable. For $a;a;a^* \leq a;a^*$ by K12. Thus, by K13, $a^*;a;a^* \leq a;a^*$. By K12, $1 \leq a^*$, so $1+(a^*;a) \leq 1+(a^*;a;a^*) \leq 1+(a;a^*) = a^*$. □

Let Σ be some finite alphabet. A Σ-term is any term constructed from elements of Σ and the symbols in the signature of Kleene algebras. These can be interpreted in the algebra $\mathbf{Reg}_\Sigma$ by the valuation R, defined as the homomorphism from the term-algebra to $\mathbf{Reg}_\Sigma$ such that $R(a) = \{a\}$ for any $a \in \Sigma$. Kozen ([15]) proves that if t and u are two Σ-terms (the variables of t and u are symbols of the alphabet Σ) then:

$$\mathsf{KC} \vdash t = u \quad \text{iff} \quad R(t) = R(u).$$

It is now easy to prove that REG and REL generate the same variety:

Theorem 2.1 (Kozen) *The following are equivalent:*

1. $\mathsf{KC} \vdash t = u$ 2. $\mathsf{REL} \models t = u$ 3. $\mathsf{REG} \models t = u$.

Proof.
From *1.* to *2.* simply show that (the universal closure of) each formula of KC holds in every relational Kleene algebra. From *2.* to *3.* note that validity of equations is preserved under subalgebras, and as any regular language model is a subalgebra of a relational Kleene algebra, we are done. Finally, we

use Kozen's result for the step from *3.* to *1.* If $t = u$ is valid in all regular language models, then it must be valid in the language model over Σ, where Σ contains all variables of t and u. In particular, $R(t) = R(u)$, so by Kozen's theorem: $\mathsf{KC} \vdash t = u$. □

2.2 PDL

Let A be a set of atomic *actions* or programs and P a set of atomic propositions. The formulas of Propositional Dynamic Logic (PDL, [21, 7]) are defined from this as follows:

$$\begin{array}{lrcl} \text{PDL-propositions:} & \phi & ::= & \bot \mid p \mid \phi \vee \phi \mid \neg\phi \mid \langle\pi\rangle\phi \\ \text{PDL-programs:} & \pi & ::= & 0 \mid 1 \mid a \mid \phi? \mid \pi + \pi \mid \pi;\pi \mid \pi^* \end{array}$$

where $p \in \mathsf{P}$ and $a \in \mathsf{A}$. Our definition is slightly nonstandard as 0 and 1 are usually not present in the language of PDL, although they are definable within the standard presentation of PDL. We add them to ensure that Kleene algebra terms may also be viewed as PDL-programs.

Other propositional connectives can be defined as usual: $\top := \neg\bot$, $\phi \wedge \psi := \neg(\neg\phi \vee \neg\psi)$, $\phi \to \psi := \neg\phi \vee \psi$, $\phi \leftrightarrow \psi := (\phi \to \psi) \wedge (\psi \to \phi)$ and $[\pi]\phi := \neg\langle\pi\rangle\neg\phi$.

PDL is interpreted in labeled transition systems (LTSs), which are models for a signature containing a binary relation-symbol R_a for every atomic action a and a unary predicate-symbol p for every atomic proposition of the same name. PDL-propositions are interpreted at the nodes of LTSs, PDL-programs are interpreted as relations between such nodes. The following defines these truth-conditions and relations simultaneously, relative to some LTS $\mathcal{M}$.

$$\begin{array}{lll} s \Vdash \bot & & \text{is never the case} \\ s \Vdash p & \text{iff} & s \in p^{\mathcal{M}} \\ s \Vdash \phi \vee \psi & \text{iff} & s \Vdash \phi \text{ or } s \Vdash \psi \\ s \Vdash \neg\phi & \text{iff} & s \nVdash \phi \\ s \Vdash \langle\pi\rangle\phi & \text{iff} & \text{there is some } t \text{ with } (s,t) \in [\![\pi]\!] \text{ and } t \Vdash \phi \end{array}$$

$$\begin{array}{lcl} [\![0]\!] & := & \emptyset \\ [\![1]\!] & := & \mathsf{id}_S \\ [\![a]\!] & := & R_a^{\mathcal{M}} \\ [\![\phi?]\!] & := & \{(s,s) \mid s \Vdash \phi\} \\ [\![\pi_1 + \pi_2]\!] & := & [\![\pi_1]\!] \cup [\![\pi_2]\!] \\ [\![\pi_1;\pi_2]\!] & := & [\![\pi_1]\!];[\![\pi_2]\!] \\ [\![\pi^*]\!] & := & [\![\pi]\!]^* \end{array}$$

Here $p^{\mathcal{M}}$ denotes the interpretation of p in $\mathcal{M}$ and similarly for $R_a^{\mathcal{M}}$. We will often denote $(s,t) \in [\![\pi]\!]$ by $s \xrightarrow{\pi} t$. We say that there is a π-*transition* from s to t. If π is a test $\phi?$ then $s = t$ and we say that $\phi?$ *succeeds* at s.

A proposition ϕ is *satisfiable* if for some node s in some LTS: $s \Vdash \phi$. A program π is satisfiable if for some LTS $[\![\pi]\!]$ is nonempty. Clearly ϕ is satisfiable iff $\phi?$ is and π is satisfiable iff $\langle\pi\rangle\top$ is.

Definition 2.2 *Let $\mathcal{M}$ and $\mathcal{N}$ be two LTSs, with domains respectively M and N, and let $Z \subseteq M \times N$. Z is a* bisimulation *between $\mathcal{M}$ and $\mathcal{N}$ (notation $Z : \mathcal{M} \underline{\leftrightarrow} \mathcal{N}$) if:*

Invariance: *Z-connected points agree on the atomic propositions: if sZt then $s \in p^{\mathcal{M}}$ iff $t \in p^{\mathcal{N}}$ for every $p \in \mathsf{P}$.*

Zig: *$sR_a^{\mathcal{M}}s'$ and sZt implies that there is a t' such that $s'Zt'$ and $tR_a^{\mathcal{N}}t'$;*

Zag: *Vice versa: if sZt and $tR_a^{\mathcal{N}}t'$ then there is an s' with $sR_a^{\mathcal{M}}s'$ and $s'Zt'$.*

Two nodes s and t are bisimilar *iff there is a bisimulation connecting the two, notation: $s \underline{\leftrightarrow} t$.* □

Lemma 2.3 *PDL-propositions are* invariant for bisimulation, *i.e. if ϕ is a PDL-proposition and $s \underline{\leftrightarrow} t$ then $s \Vdash \phi$ iff $t \Vdash \phi$.*

Proof.
Simultaneously prove that PDL-programs are *safe* for bisimulation: if π is a PDL-program and Z is any bisimulation then sZt and $s \xrightarrow{\pi} s'$ implies that there is a t' such that $t \xrightarrow{\pi} t'$ and tZt'. □

An important class of LTSs is that of the *unraveled* ones. These are LTSs $\mathcal{M}$ such that:

1. $R := \bigcup_{a \in \mathsf{A}} R_a^{\mathcal{M}}$ is well-founded: there exists no infinite backward chain of R-edges.
2. $\mathcal{M}$ is *rooted* in some element s: every $t \in \mathcal{M}$ can be reached from s via a finite number of R-steps.
3. Every element except the root has a unique R-predecessor. The root has no predecessor.
4. For every distinct atomic actions $a, b \in \mathsf{A}$, the transitions $R_a^{\mathcal{M}}$ and $R_b^{\mathcal{M}}$ are disjoint.

In other words, an unraveled LTS is a rooted intransitive tree with all transitions disjoint.

Lemma 2.4 *Any satisfiable ϕ may be satisfied in the root of an unraveled LTS. Consequently, for any satisfiable program π there is an unraveled LTS with a π-transition from the root.*

Proof.
This is due to the facts that PDL-propositions are invariant for bisimulation and that any node in an LTS is bisimilar to the root of an unraveled LTS. □

A proposition is *valid* iff its negation is not satisfiable, i.e. if it is satisfied at every node in every LTS. The Segerberg axioms ([27]) together with axioms for 0 and 1, displayed in the following table, provide a finite axiomatization of PDL-validity. A perspicuous completeness proof may also be found in [8].

Axioms:

P1	All instances of all tautologies of propositional logic.	
P2	$\langle\pi\rangle(\phi \vee \psi) \leftrightarrow (\langle\pi\rangle\phi \vee \langle\pi\rangle\psi)$	(distribution)
P3	$\langle\pi_1; \pi_2\rangle\phi \leftrightarrow \langle\pi_1\rangle\langle\pi_2\rangle\phi$	(composition)
P4	$\langle\pi_1 + \pi_2\rangle\phi \leftrightarrow (\langle\pi_1\rangle\phi \vee \langle\pi_1\rangle\phi)$	(union)
P5	$\langle\phi?\rangle\psi \leftrightarrow (\phi \wedge \psi)$	(test)
P6	$(\phi \vee \langle\pi\rangle\langle\pi^*\rangle\phi) \rightarrow \langle\pi^*\rangle\phi$	(iteration)
P7	$[\pi^*](\phi \rightarrow [\pi]\phi) \rightarrow (\phi \rightarrow [\pi^*]\phi)$	(induction)
P8	$[0]\bot$	(zero)
P9	$\langle 1\rangle\phi \leftrightarrow \phi$	(identity)

Rules:

MP	$\phi, \phi \rightarrow \psi \;/\; \psi$	(modus ponens)
G	$\phi \;/\; [\pi]\phi$	(generalization)
Sub	$\phi \leftrightarrow \psi \;/\; \chi[\phi] \leftrightarrow \chi[\psi]$	(substitution)

The Sub-rule should be interpreted as follows. If ϕ has been proved equivalent to ψ and χ contains some occurrence of ϕ then χ is equivalent to a formula where this occurrence has been replaced by ψ. We write PDL $\vdash \phi$ iff ϕ is provable using the above rules and axioms. Completeness states that PDL $\vdash \phi$ iff ϕ is a valid proposition.

PDL has the finite model property: any satisfiable PDL-formula is satisfiable in a finite LTS. Thus restricting to finite LTSs only gives us the same theory (as opposed to, say, first order logic). In fact, an extension of PDL, known as the modal μ-calculus ([14]), is also known to have this property (see [28]). Together with the axiomatization above, this immediately gives us decidability of validity. However, we know more: it is decidable in deterministic exponential time ([7]). Again, this may be deduced from the embedding (in linear time) into the modal μ-calculus, which is exponential time complete. The algebras we will deal with in this paper have a close connection to PDL, so that results known for PDL will sometimes carry over to this algebraic setting.

2.3 Dynamic algebras

Dynamic algebras ([13, 22, 24]) are basically an algebraic approach to PDL. As PDL is a two-sorted logic (with greatest emphasis on the sort of propositions), dynamic algebras are two sorted as well.

Given any (possibly empty) set S, $\mathsf{DA}(\mathrm{S})$, the dynamic algebra over S, is defined as $(\mathsf{K}, \mathsf{B}, \Diamond)$ where:

- K is the relational Kleene algebra with as domain the set of all binary relations on S.
- B is the boolean algebra (say of signature $\{\bot, \neg, \vee\}$: bottom, complement and join) with as domain the powerset of S and the operations interpreted as the empty set, set-complement with respect to S and union, respectively.
- $\Diamond$ (what Pratt ([24]) calls the *enables* operator) is a function from $\mathsf{K} \times \mathsf{B}$ to B. We will often write $\langle R \rangle X$ instead of $\Diamond(R, X)$. It is defined as:

$$\langle R \rangle X := \{s \in S \mid \exists t \in X. sRt\}$$

DA, the class of all dynamic algebras consists of the dynamic algebras over all sets:

$$\mathsf{DA} := \{\mathsf{DA}(S) \mid S \text{ is a set}\}.$$

Note that we may use the set A of atomic actions as program (Kleenean) variables in dynamic algebra and the set of atomic propositions P as variables of the boolean sort. Then terms of dynamic algebra coincide exactly with the propositions and test-free programs of PDL.

Any axiomatization of this algebra calls for both equations relating program terms and equations relating boolean ones. The latter may contain, besides the boolean symbols, the enables operator $\Diamond$. We get an axiomatization of the valid equations of the boolean sort by translating the Segerberg axioms for PDL (except the one for tests of course) into algebraic format:

S0	Axioms of boolean algebra	
S1	$\langle a \rangle \bot = \bot$	(normality)
S2	$\langle a \rangle (p \vee q) = \langle a \rangle p \vee \langle a \rangle q$	(additivity)
S3	$\langle 0 \rangle p = \bot$	(zero)
S4	$\langle 1 \rangle p = p$	(identity)
S5	$\langle a + b \rangle p = \langle a \rangle p \vee \langle b \rangle p$	(plus)
S6	$\langle a; b \rangle p = \langle a \rangle \langle b \rangle p$	(composition)
S7	$\langle a^* \rangle p = p \vee \langle a \rangle \langle a^* \rangle p$	(iteration)
S8	$\langle a^* \rangle p = p \vee \langle a^* \rangle (\neg p \wedge \langle a \rangle p)$	(induction)

To see that this set of axioms is complete, note that the completeness proof of the PDL-axioms (see the original [27] or the exposition in [8]) does not hinge on the fact that tests may occur in standard PDL-programs.

In most presentations of dynamic algebra, the above equations define the notion of dynamic algebra: any algebra satisfying S0-8 is deemed a dynamic algebra. This imposes no structure whatsoever on the program sort. However, this is dealt with by introducing the notion of *separability*. A two-sorted algebra $(\mathsf{K}, \mathsf{B}, \Diamond)$ satisfying S0-8 is separable iff the following holds:

$$\text{(Sep)} \quad \forall a, b \in \mathsf{K}. (a \neq b \to \exists p \in \mathsf{B}. \langle a \rangle p \neq \langle b \rangle p).$$

Sep together with S0-8 gives us an axiomatization of dynamic algebras in our sense: the class of algebras satisfying S0-8 and separability generates the same variety as the class of dynamic algebras (theorem 6.4, [24]). What we refer to as dynamic algebras has been referred to as *full Kripke structures* in [24]. Subalgebras of dynamic algebras are referred to as (just) Kripke structures in this same article and as *dynamic set algebras* in [18].

What about the valid program equations? As the program terms of dynamic algebra cannot refer to $\Diamond$, nor in fact to any of the boolean structure, a program-equation is valid in all dynamic algebras

iff it is valid in all Kleene algebras. Thus we get some results for free: dynamic algebra is not finitely axiomatizable (equationally) and we have a whole plethora of axiomatizations to choose from on the program-side. By the remarks on separability, adding Sep to the boolean sort equations S0-8 is also an axiomatization, albeit a nonstandard one from an algebraic point of view.

Remark: S0-8 are complete by themselves with respect to valid equalities between test-free PDL-propositions. If S0-8 are taken in combination with some axiomatization of Kleene algebra, then S7 becomes redundant, by the fact that $a^* = 1 + (a; a^*)$ (K12) is valid in Kleene algebras:

$$\begin{array}{lcll} \langle a^* \rangle p & = & \langle 1 + (a; a^*) \rangle p & \\ & = & \langle 1 \rangle p \vee \langle a; a^* \rangle p & \text{(S5)} \\ & = & p \vee \langle a \rangle \langle a^* \rangle p & \text{(S4, S6)} \end{array}$$

Similarly, the boolean law of idempotency of $\vee$ (contained in S0) also becomes derivable:

$$\begin{array}{lcll} p \vee p & = & \langle 1 \rangle p \vee \langle 1 \rangle p & \text{(S4)} \\ & = & \langle 1 + 1 \rangle p & \text{(S5)} \\ & = & \langle 1 \rangle p & \\ & = & p & \text{(S4)} \end{array}$$

□

2.4 Test algebras

Test algebras ([24, 18, 29]) relieve us of the restriction that programs be test-free. A test algebra consists of a dynamic algebra $(\mathsf{K}, \mathsf{B}, \diamond)$ over some set S, together with a function $? : \mathsf{B} \to \mathsf{K}$ defined as follows:

$$X? := \{(s, s) \mid s \in X\}$$

where ? is written in postfix notation. The resulting algebra will be denoted by $\mathsf{TA}(S)$. The class of all such algebras is denoted by TA.

An axiomatization of test algebras is given by the following ingredients:

- An axiomatization of Kleene algebras, for instance K1-14.
- The axiomatization of the boolean part of dynamic algebras, consisting of S0-S8.
- The following additional axioms:

T1	$\langle p? \rangle q = p \wedge q$	(test diamond)
T2	$\bot? = 0$	(bottom test)
T3	$(p \vee q)? = p? + q?$	(test sum)
T4	$(p \wedge q)? = p?; q?$	(test composition)
T5	$(\langle a \rangle \top)?; a = a$	(domain test)

Let TC (Test Calculus) denote the resulting calculus.

There are some redundancies in this system: S1-4 are now derivable.

S1: First note that $\langle a \rangle p = \langle a; p? \rangle \top$, by reasoning as follows:

$$\begin{array}{lcll} \langle a \rangle p & = & \langle a \rangle (p \wedge \top) & \\ & = & \langle a \rangle \langle p? \rangle \top & \text{(T1)} \\ & = & \langle a; p? \rangle \top & \text{(S6)} \end{array}$$

We will use this freely in the proofs that follow.

Now S1 follows by the following reasoning:

$$\begin{array}{lcll} \langle a\rangle\bot & = & \langle a;\bot?\rangle\top & \\ & = & \langle a;0\rangle\top & \text{(T2)} \\ & = & \langle 0\rangle\top & \\ & = & \langle\bot?\rangle\top & \text{(T2)} \\ & = & \bot\wedge\top & \text{(T1)} \\ & = & \bot & \end{array}$$

S2:

$$\begin{array}{lcll} \langle a\rangle(p\vee q) & = & \langle a;(p\vee q)?\rangle\top & \\ & = & \langle a;(p?+q?)\rangle\top & \text{(T3)} \\ & = & \langle(a;p?)+(b;q?)\rangle\top & \\ & = & \langle a;p?\rangle\top\vee\langle a;q?\rangle\top & \text{(S5)} \\ & = & \langle a\rangle p\vee\langle a\rangle q & \end{array}$$

S3:

$$\begin{array}{lcll} \langle 0\rangle p & = & \langle\bot?\rangle p & \text{(T2)} \\ & = & \bot\wedge p & \text{(T1)} \\ & = & \bot & \end{array}$$

S4: First we prove that $\top? = 1$ is derivable:

$$\begin{array}{lcll} \top? & = & \top?;1 & \\ & = & \top?;(\langle 1\rangle\top)?;1 & \text{(T5)} \\ & = & (\top\wedge\langle 1\rangle\top)?;1 & \text{(T4)} \\ & = & (\langle 1\rangle\top)?;1 & \\ & = & 1 & \text{(T5)} \end{array}$$

Now S4 follows: $\langle 1\rangle p = \langle\top?\rangle p = \top\wedge p = p$.

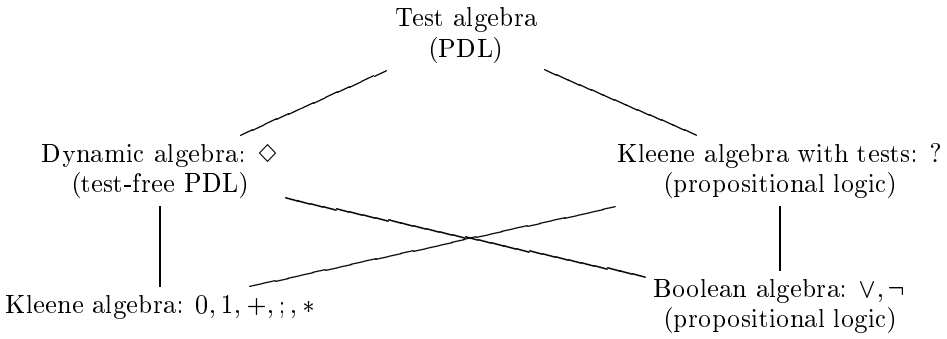
Summarizing, we claim that a complete system for test algebra is given by the axioms of Kleene algebra and those of boolean algebra, together with the extra axioms S5, S6, S8, T1-5.

If we use A as our set of program variables and P as our boolean variables, then there is a one-one correspondence between test algebra terms and PDL-formulas. When it is harmless to do so, we confuse the two. This correspondence already gives us that test algebra has the *finite algebra property*: any invalid equation may be falsified in a finite test algebra. Simply use the finite model property of PDL to prove this, noting that a program-equation $t = u$ is valid iff $\langle t\rangle p \leftrightarrow \langle u\rangle p$ is valid, where p does not occur in t nor in u. Furthermore, we also have that validity in test algebra is decidable in deterministic exponential time (as a PDL-formula ϕ is valid iff the equation $\phi? = \top$ is valid).

Again, we diverge from the standard nomenclature in the literature, which refers to algebras satisfying S0-8 and T1 as test algebras. If only separable algebras satisfying these equations are considered, we obtain the same generated variety as that of TA (this follows from Corollary 1, [29], using the finite algebra property). Subalgebras of test algebras are referred to as *Kripke test structures* in [29] and as *dynamic test set algebras* in [18].

We finish this section by summing up. The following figure positions the algebras we mentioned within a landscape: moving up in the landscape means adding the mentioned operators to the language. Algebras that have a boolean component also have a corresponding logic: it is the logic of boolean-type formulas ϕ for which $\phi = \top$ is a valid equation. To make the landscape more symmetric we also added *Kleene algebra with tests* ([16]), which may be viewed as we present it here: two-sorted algebras with a Kleenean and a boolean component but with only ? as an interacting operator. As

this only enriches the set of program terms, the logic remains standard propositional logic.



3 Completeness

3.1 The boolean side

For completeness of TC with respect to equations of the boolean sort, we will refer to completeness of PDL. In fact, for completeness on the boolean side, we do not need all of TC. A finite equational part of it suffices. So let TC_0 denote the system consisting of S0-S8 together with T1. We prove that for boolean terms t and u, $\models t = u$ ($t = u$ is valid in all test algebras) iff this equation is derivable from TC_0. This fact was already proved using the free algebra construction in [29] (Corollary 2 of that paper).

The main lemma we will use is:

Lemma 3.1 $\mathsf{PDL} \vdash \phi$ *implies* $\mathsf{TC}_0 \vdash \phi = \top$.

Proof.
Prove this using induction on PDL-proofs. □

Theorem 3.2 TC_0 *is complete with respect to the valid test algebra equations of the boolean sort.*

Proof.
Soundness is obvious. For completeness we reason as follows. If t and u are test algebra terms of the boolean sort such that $t = u$ is valid, then it is easy to see that $t \leftrightarrow u$ must be a valid PDL-formula. By the above lemma $\mathsf{TC}_0 \vdash (t \leftrightarrow u) = \top$. From this it easily follows that $\mathsf{TC}_0 \vdash t = u$. □

As a corollary we get completeness of TC with respect to unsatisfiable program terms.

Corollary 3.3 $\mathsf{TA} \models t = 0$ *iff* $\mathsf{TC} \vdash t = 0$.

Proof.
Suppose t is interpreted as the empty relation in any test algebra. Then $\langle t \rangle \top = \bot$ is valid, hence derivable. By T5, $t = (\langle t \rangle \top)?; t$. So t is provably equal to $\bot?; t$, which is equal to $0; t$ by T2, hence to 0, by K4. □

3.2 The Kleenean side

In this subsection we prove completeness for program-equations. The proof is based on the completeness proof of [16].

Fix two program terms t_1 and t_2. For the remainder of this section, we restrict the set of variables to those occurring in either t_1 or t_2. Thus we consider only a finite number of variables. These

variables are of two sorts: boolean variables and program variables. These will also be referred to as atomic propositions and atomic actions respectively.

Let:

$$\Gamma := \{\phi \mid \ \phi? \text{ is a subterm of either } t_1 \text{ or } t_2\}$$

Definition 3.4 *The* Fischer-Ladner closure of Γ *(*FL-closure of Γ*, see [7]) is the smallest set S such that for all $a \in \mathsf{A}$, all PDL-programs π, π_1, π_2 and all PDL-propositions ϕ and ψ:*

$$\begin{array}{lll}
\phi \in \Gamma & \textit{implies} & \phi \in S \\
\neg\phi \in S & \textit{implies} & \phi \in S \\
\phi \vee \psi \in S & \textit{implies} & \phi, \psi \in S \\
\langle\pi\rangle\phi \in S & \textit{implies} & \phi \in S \\
\langle\pi_1;\pi_2\rangle\phi \in S & \textit{implies} & \langle\pi_1\rangle\langle\pi_2\rangle\phi \in S \\
\langle\pi_1 \cup \pi_2\rangle\phi \in S & \textit{implies} & \langle\pi_1\rangle\phi, \langle\pi_2\rangle\phi \in S \\
\langle\pi^*\rangle\phi \in S & \textit{implies} & \langle\pi\rangle\langle\pi^*\rangle\phi \in S \\
\langle\phi?\rangle\psi \in S & \textit{implies} & \phi \in S
\end{array}$$

The FL-closure of Γ will be denoted by FL(Γ). □

The Fischer-Ladner closure of a finite set such as Γ is again a finite set of formulas (see for instance [8]), of size linear in the number of symbols in Γ. This set is closed under subformulas. Enumerate FL(Γ) as $\gamma_1, \ldots, \gamma_k$. Define the set of *guards* $\mathbb{G}$ as:

$$\{(\phi_1 \wedge \ldots \wedge \phi_k)? \mid \forall i.\phi_i \in \{\gamma_i, \neg\gamma_i\} \text{ and } \phi_1 \wedge \ldots \wedge \phi_k \text{ is satisfiable}\}$$

We let (subscripted) initial greek letters $\alpha, \alpha_1, \beta, \ldots$ range over guards. If α is the guard $(\phi_1 \wedge \ldots \wedge \phi_k)?$ then we write $\phi \in \alpha$ if ϕ is one of the conjuncts ϕ_i $(1 \leq i \leq k)$. We say ϕ *occurs in* α.

The set of *guarded strings* $\mathbb{GS}$ is the set of *satisfiable* program terms of the following form:

$$\alpha_0; a_1; \alpha_1; \ldots; a_n; \alpha_n$$

where each α_i is a guard and each a_i is a program variable. The *length* of the above guarded string is n. A guarded string of length 0 is then simply a guard. We will often denote an arbitrary guarded string by $\alpha;\sigma;\beta$, $\alpha;\sigma$ or $\sigma;\alpha$ to be able to speak about the outer guards. None of these notations is meant to imply that the length of the guarded string must be of length greater than 0. In case $\alpha;\sigma;\beta$ is used to indicate a single guard, α and β coincide and σ is empty.

Guarded strings that match their guards are always compatible: if $\sigma;\alpha$ and $\alpha;\tau$ are guarded strings then $\sigma;\alpha;\tau$ is of the right form. Moreover it is satisfiable. To show this we need the notion of Γ-*bisimulation*.

Definition 3.5 *Two points s and t in an LTS are* Γ-equivalent *iff they agree on all formulas in FL(Γ). In other words: they are Γ-equivalent iff the same guard α succeeds at s and t. Two points s and t are* Γ-bisimilar *if:*

Invariance: *They agree on all atomic propositions (by assumption these occur in FL(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 3.6 *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 FL(Γ). The base cases for $\bot$ 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 $s \xrightarrow{\pi} s$. We prove that $t \xrightarrow{\pi} t$ also holds, by induction on the size of π.

- π cannot be an atomic action, as $\mathcal{M}$ is unraveled.
- If $s \xrightarrow{\psi?} s$ then by our assumption on tests in π there is also a $\psi?$-loop on t.
- π cannot be 0, as $s \xrightarrow{0} s$ never holds. If $\pi = 1$ then the claim obviously holds, as $t \xrightarrow{1} t$ for any t.
- Let $\pi = \pi_1; \pi_2$. Then $s \xrightarrow{\pi_1} s' \xrightarrow{\pi_2} s$ for some s'. So s' must be reachable from s and vice versa. As $\mathcal{M}$ is unraveled and hence contains no cycles, s must be equal to s'. So we may use the induction hypothesis: $t \xrightarrow{\pi_1} t \xrightarrow{\pi_2} t$.
- Let $\pi = \pi_1 + \pi_2$. Then $s \xrightarrow{\pi_i} s$ for some $i \in \{1, 2\}$. Use the induction hypothesis.
- Let $\pi = \pi_1^*$. Then clearly $t \xrightarrow{\pi} t$.

In the next claim we again assume that s and t agree on all tests in π.

Claim 2: For any ϕ such that $\langle\pi\rangle\phi \in$ FL(Γ), if $s \xrightarrow{\pi} s' \Vdash \phi$ with $s \neq s'$ then $t \Vdash \langle\pi\rangle\phi$. The proof proceeds by induction on π:

- If π is an atomic action a then by Γ-bisimilarity there is a t' such that $t \xrightarrow{a} t'$ and t' is Γ-equivalent to s', hence $t' \Vdash \phi$.
- As $s \neq s'$, π cannot be a test. Likewise, π cannot be 1. For even more obvious reasons π cannot be 0.
- If $\pi = \pi_1; \pi_2$ then $s \xrightarrow{\pi_1} s'' \xrightarrow{\pi_2} s'$ for some s''. If $s = s''$ then we have already proved that $t \xrightarrow{\pi_1} t$. Use the induction hypothesis on π_2 to show that $t \Vdash \langle\pi_2\rangle\phi$, hence $t \Vdash \langle\pi_1; \pi_2\rangle\phi$. If $s \neq s''$ then use the induction hypothesis on π_1 to show that $t \Vdash \langle\pi_1; \pi_2\rangle\phi$.
- The case of sum is trivial.
- If $\pi = \pi_1^*$ then $s = s_0 \xrightarrow{\pi_1} s_1 \xrightarrow{\pi_1} s_2 \ldots \xrightarrow{\pi_1} s_n = s'$. As $s \neq s'$, for some $i < n$: $s = s_i \neq s_{i+1}$. $\langle\pi_1^*\rangle\phi$ is satisfied in s_{i+1} so we may use the induction hypothesis on π_1 (applied to the formula $\langle\pi_1^*\rangle\phi$) to show that $t \Vdash \langle\pi_1\rangle\langle\pi_1^*\rangle\phi$, hence $t \Vdash \langle\pi_1^*\rangle\phi$.

We continue with the proof of the lemma. Suppose $s \Vdash \langle\pi\rangle\phi$ for some $\langle\pi\rangle\phi \in$ FL(Γ). 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 3.7 *If $\alpha; a; \beta$ and $\beta; \tau$ are two guarded strings then $\alpha; a; \beta; \tau$ is also a guarded string.*

Proof.
We need to prove that $\alpha; a; \beta; \tau$ is satisfiable. By assumption $\alpha; a; \beta$ and $\beta; \tau$ are satisfiable. Let $\mathcal{M}$ be an unraveled LTS rooted in s such that α succeeds on s and there is an a-step from s to s' where β succeeds. Let $\mathcal{N}$ be an unraveled LTS such that there is a $\beta; \tau$-transition emanating from the root

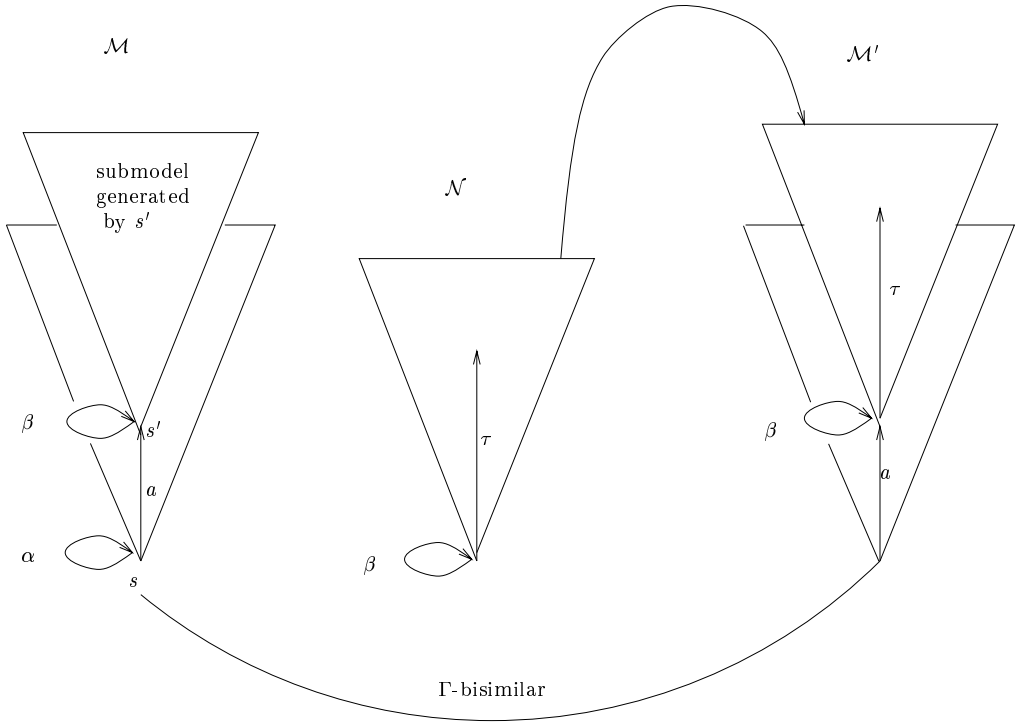


Figure 1: How to extend a guarded string.

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 3.6 this implies Γ-equivalence, so α succeeds at the root of $\mathcal{M}'$. So there is an $\alpha; a; \beta$-step from the root of $\mathcal{M}'$ to the root of $\mathcal{N}$, from which we may continue with a τ-transition. So $\alpha; a; \beta; \tau$ is satisfiable in $\mathcal{M}'$. The situation is described in figure 1. □

Corollary 3.8 *Guarded strings are closed under* coalesced product*: if $\sigma; \alpha$ and $\alpha; \tau$ are two guarded strings then $\sigma; \alpha; \tau$ is a guarded string.*

Proof.
Repeatedly apply lemma 3.7. □

We make an algebra $\mathcal{G}$ out of guarded strings of the test algebra signature.

- The Kleenean domain consists of all subsets of $\mathbb{GS}$ where:
  - 0 is interpreted as the empty set.
  - 1 is interpreted as $\mathbb{G}$.
  - $R; S := \{\sigma; \alpha; \tau \mid \alpha \in \mathbb{G}, \sigma; \alpha \in R, \alpha; \tau \in S\}$.
  - $+$ is interpreted by union.
  - $R^* = \bigcup_{n \geq 0} A^n$ where $R^0 = \mathbb{G}$ and $R^{n+1} = R; R^n$.
- The boolean domain consists of all subsets of $\mathbb{G}$. Disjunction and negation are interpreted by union and set-complement respectively.
- The enables operator is interpreted as follows. If $R \subseteq \mathbb{GS}$ and $X \subseteq \mathbb{G}$ then

$$\langle R \rangle X := \{\alpha \in \mathbb{G} \mid \exists \beta \in X. \, \exists \sigma. \; \alpha; \sigma; \beta \in R\}$$

  Note that if $\alpha \in R \cap X$ then $\alpha \in \langle R \rangle X$.

- ? is interpreted by the identity. This is possible as the boolean domain is a subset of the Kleenean domain.

Theorem 3.9 *$\mathcal{G} \subseteq \mathcal{A}$ for some test algebra $\mathcal{A}$. Hence $\mathcal{G} \models \mathsf{TC}$.*

Proof.
Let $\mathcal{A}$ be the test algebra over $\mathbb{GS}$, considered as a set. That is: $\mathcal{A} = \mathsf{TA}(\mathbb{GS})$ (see section 2.4). We will provide an embedding of $\mathcal{G}$ into $\mathcal{A}$. Because test algebras are two-sorted, this embedding is given by a function k from $\wp(\mathbb{GS})$ to binary relations over $\mathbb{GS}$:

$$k(R) := \{(\sigma;\alpha\ ,\ \sigma;\alpha;\tau) \in \mathbb{GS}^2 \mid \alpha \in \mathbb{G},\ \alpha;\tau \in R\}$$

and a function b from $\wp(\mathbb{G})$ to $\wp(\mathbb{GS})$:

$$b(X) := \{\sigma;\alpha \in \mathbb{GS} \mid \alpha \in X\}.$$

We prove that these two functions together constitute an embedding.

- k is injective, for if $\alpha;\sigma \in R$ but $\alpha;\sigma \notin S$ then $(\alpha, \alpha;\sigma)$ is a transition in $h(R)$ but not in $h(S)$.
- $k(\emptyset) = \emptyset$. So k preserves zero.
- $k(\mathbb{G})$ is the identity on $\mathbb{GS}$. So k preserves the identity.
- That k distributes over $+$ and ; is left as an exercise.
- Note that k distributes over arbitrary unions, so in order to prove that $k(R^*) = k(R)^*$ it suffices to show that $k(R^n) = k(R)^n$, by induction on n. The base case proceeds by noticing that k preserves the identity, while the inducton step uses the fact that k distributes over ;.
- We move to the boolean sort. b is also injective, for if $\alpha \in X \setminus Y$ then $\alpha \in b(X)$ but $\alpha \notin b(Y)$. It is trivial to verify that b distributes over disjunction and negation (which in both algebras correspond to union and set-complement respectively).
- We must prove that $b(\langle R\rangle X) = \langle k(R)\rangle b(X)$. So suppose that $\sigma;\alpha \in b(\langle R\rangle X)$. Then by definition of b, $\alpha \in \langle R\rangle X$, i.e. for some $\beta \in X$ there exists a guarded string of the form $\alpha;\tau;\beta \in R$. Hence $(\sigma;\alpha, \sigma;\alpha;\tau;\beta)$ is a $k(R)$-transition and $\sigma;\alpha;\tau;\beta$ is an element of $b(X)$. Thus $\sigma;\alpha \in \langle k(R)\rangle b(X)$. The other direction is proved similarly.
- Finally, we need that $k(X?) = b(X)?$. $k(X?) = k(X)$ can easily be seen to be $\{(\sigma;\alpha, \sigma;\alpha) \mid \alpha \in X\}$ as X contains only guards. This gives us the statement. □

We provide $\mathcal{G}$ with a valuation $[\![\_]\!]$ to variables. On boolean variables p,

$$[\![p]\!] = \{\alpha \in \mathbb{G} \mid p \in \alpha\}.$$

On program variables a,

$$[\![a]\!] = \{\alpha;a;\beta \mid \alpha;a;\beta \text{ is satisfiable, } \alpha, \beta \in \mathbb{G}\ \}.$$

This interpretation can be homomorphically extended to all terms. Note then that for any term t, $[\![t]\!]$ is a set of guarded strings, which are nothing but special kinds of program terms.

Lemma 3.10 *For $\phi \in FL(\Gamma)$: $[\![\phi]\!] = \{\alpha \in \mathbb{G} \mid \phi \in \alpha\}$.*

Proof.
If ϕ is an atomic proposition, this is just by definition. If $\phi = \bot$ we need to prove that $\{\alpha \mid \bot \in \alpha\}$ is empty. This is true by the fact that guards are assumed to be satisfiable. For the negation case we have to show that for any guard α, $\phi \notin \alpha$ iff $\neg\phi \in \alpha$ (if $\neg\phi \in \mathrm{FL}(\Gamma)$), which is again due to the fact that guards are satisfiable. The disjunction case hinges on the fact that $\phi \vee \psi \in \alpha$ iff $\phi \in \alpha$ or $\psi \in \alpha$ (when $\phi \vee \psi \in \mathrm{FL}(\Gamma)$), again trivial to prove.

The difficulty lies of course in the modality case $\langle\pi\rangle\phi$. The induction hypothesis tells us that for any formula ψ whose test $\psi?$ occurs in π the lemma has already been proved. We prove the following:

Claim: Suppose t is a term built up from program variables, tests $\psi?$ such that $\psi \in \mathrm{FL}(\Gamma)$ is a formula for which the lemma has already been proved, and the Kleene operators $0, 1, +, ;, *$. Note that $[\![t]\!]$ is a possibly infinite set of guarded strings, which are just special kinds of PDL-programs. Given a test algebra and an assignment to variables, PDL-programs denote binary relations. It thus makes perfect sense to consider $\sum[\![t]\!]$, the infinite sum of the programs in $[\![t]\!]$. Given a test algebra and an assignment $\sum[\![t]\!]$ denotes the union of all binary relations denoted by the programs in $[\![t]\!]$. We prove that $\sum[\![t]\!] = t$ is valid in all test algebras, that is: the relation we described always coincides with the relation denoted by t.

Proof of Claim: We prove it by induction on t.

- $\sum[\![a]\!] = a$, as whenever $s \xrightarrow{a} t$ then some guard α succeeds at s and some guard β succeeds at t, which makes $\alpha; a; \beta$ a satisfiable element of $[\![a]\!]$.
- We assume the lemma has been proved for $\psi \in \mathrm{FL}(\Gamma)$. Thus $\sum[\![\psi?]\!] = \sum\{\alpha \in \mathbb{G} \mid \psi \in \alpha\}$. Whenever $\psi?$ succeeds at some node s then some guard α succeeds at s. As $\psi \in \mathrm{FL}(\Gamma)$, $\psi \in \alpha$. So: $\sum[\![\psi?]\!] = \psi?$ is valid in test algebras.
- The 0-case is trivial, as the empty sum is $\emptyset$.
- The 1-case proceeds as in the test-case: at every node some guard succeeds, so $\sum \mathbb{G} = 1$ is valid.
- The $+$-case is trivial: $\sum([\![t]\!] \cup [\![u]\!]) = \sum[\![t]\!] + \sum[\![u]\!]$.
- For the ;-case we prove that $\sum([\![t]\!]; [\![u]\!]) = \sum[\![t]\!]; \sum[\![u]\!]$. Note that ; on the lefthand side of the equation is as defined in the algebra $\mathcal{G}$, while ; on the right is the usual relational composition.

  From left to right, suppose that $\sigma; \alpha \in [\![t]\!]$ and $\alpha; \tau \in [\![u]\!]$ (and hence $\sigma; \alpha; \tau \in [\![t]\!]; [\![u]\!]$) such that $s_1 \xrightarrow{\sigma} s_2 \xrightarrow{\tau} s_3$, where α succeeds on s_2. Then there is a $\sigma; \alpha$-transition from s_1 to s_2 and a $\alpha; \tau$-transition from s_2 to s_3, so $(s_1, s_2) \in \sum[\![t]\!]$ and $(s_2, s_3) \in \sum[\![u]\!]$.

  From right to left, suppose $\sigma; \alpha \in [\![t]\!]$ and $\beta; \tau \in [\![u]\!]$ such that there is a $\sigma; \alpha$-transition from s_1 to s_2 and a $\beta; \tau$-transition from s_2 to s_3. As both α and β succeed on s_2, they must be compatible, hence equal. So there is a $\sigma; \alpha; \tau$-transition from s_1 to s_3. As $\sigma; \alpha; \tau \in [\![t]\!]; [\![u]\!]$ we are done.
- For the $*$-case, we use the cases we have already dealt with for 1 and ; to show that $\sum([\![t]\!]^n) = (\sum[\![t]\!])^n$ for any n. Then $\sum[\![t^*]\!] = \sum(\bigcup_{n\geq 0}[\![t]\!]^n) = \sum_{n\geq 0}(\sum([\![t]\!]^n)) = \sum_{n\geq 0}(\sum[\![t]\!])^n = \sum_{n\geq 0} t^n$ by the induction hypothesis. This latter expression is equivalent to t^* in test algebras so we are done.

This completes the proof of the Claim. we continue with the proof of the lemma itself. Recall that we are dealing with the case $\langle\pi\rangle\phi$. We need to prove that:

$$\exists \alpha; \sigma; \beta \in [\![\pi]\!] \text{ and } \phi \in \beta \quad \text{iff} \quad \langle\pi\rangle\phi \in \alpha$$

where we may assume that if $\psi?$ is a test in π then the lemma has already been proved for ψ. Note that π is then a term to which our Claim applies.

From left to right, if $\alpha; \sigma; \beta$ is a guarded string in $[\![\pi]\!]$ such that $\phi \in \beta$ then, as guarded strings are satisfiable there exist nodes s_1 and s_2 such that $s_1 \xrightarrow{\sigma} s_2$, α succeeds at s_1 and β succeeds at s_2

(hence $s_2 \Vdash \phi$). By the Claim there must also be a π-transition from s_1 to s_2. Now if $\langle\pi\rangle\phi \notin \alpha$ then $\neg\langle\pi\rangle\phi \in \alpha$, which contradicts the above.

For the other direction, suppose α is a guard containing the proposition $\langle\pi\rangle\phi$. As α is satisfiable there exist nodes s_1 and s_2 such that $s_1 \xrightarrow{\pi} s_2 \Vdash \phi$ with α succeeding at s_1. By the Claim there is some guarded string $\beta;\sigma;\gamma$ in $[\![\pi]\!]$ such that $s_1 \xrightarrow{\beta} s_1 \xrightarrow{\sigma} s_2 \xrightarrow{\gamma} s_2$. But then $\alpha = \beta$ and $\phi \in \gamma$, so we are done. □

Corollary 3.11 *If α is a guard then $[\![\alpha]\!] = \{\alpha\}$.*

Proof.
Suppose $\alpha = (\phi_1 \wedge \ldots \wedge \phi_k)?$. By definition $\phi_i \in \alpha$ for each i. By the above lemma $\alpha \in [\![\phi_i]\!]$ for each i, hence $\alpha \in [\![\phi_1 \wedge \ldots \wedge \phi_k]\!] = [\![\alpha]\!]$. To show that $[\![\alpha]\!]$ contains *only* α, suppose $\beta \in [\![\phi_1 \wedge \ldots \wedge \phi_k]\!]$. Then $\beta \in [\![\phi_i]\!]$ for each i, hence $\phi_i \in \beta$ by the lemma. This implies that $\alpha = \beta$. □

Consider the alphabet Σ consisting of all program variables and all guards. By the initial assumption of this section, Σ must be a finite set. Let $\mathbf{Reg}_\Sigma$ be the algebra of regular sets over this alphabet (see section 2.1). Such an algebra comes equipped with an interpretation R for Σ-terms. Note that Σ-terms can also serve as input to $[\![\_]\!]$.

Definition 3.12 *Define an* externally guarded term *to be either a guard or a term of the form $\alpha;t;\beta$ where $\alpha, \beta \in \mathbb{G}$.* □

Theorem 3.13 *Let t be a term constructed from program variables, terms $\phi?$ for ϕ in the FL-closure of Γ and the Kleene algebra operators. Then there is a $\widehat{t}$ such that:*

1. *$\widehat{t}$ is a finite set of externally guarded Σ-terms.*
2. *$\mathsf{TC} \vdash t = \sum \widehat{t}$.*
3. *For every $\sigma \in \widehat{t}$: $R(\sigma) = [\![\sigma]\!]$.*

Proof.
The third condition may mystify the reader, as $R(\sigma)$ will contain strings over Σ, while $[\![\sigma]\!]$ will contain guarded strings, which are basically just certain kinds of program terms. However, if we denote string concatenation by ; and read guarded strings modulo associativity of ; then the problem disappears.

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

- $\widehat{0} = \emptyset$. As $\sum\emptyset$ is 0 by definition, the conditions are trivial to verify.
- $\widehat{1} = \mathbb{G}$. We check the conditions.
  1. Let $\alpha_1, \ldots, \alpha_n$ enumerate $\mathbb{G}$. These are tests of boolean terms. Let $\psi_1, \ldots, \psi_n$ be these formulas. Now $\models \psi_1 \vee \ldots \vee \psi_n = \top$, so by theorem 3.2, $\mathsf{TC}_0 \vdash \psi_1 \vee \ldots \vee \psi_n = \top$. But then $\mathsf{TC}_0 \vdash (\psi_1 \vee \ldots \vee \psi_n)? = \top?$. The left term is equivalent, by T3, to $\alpha_1 + \ldots + \alpha_n$, while the right is equivalent to 1, by T3.
  2. By corollary 3.11 $R(\alpha) = \{\alpha\} = [\![\alpha]\!]$ for any guard α.
- $\widehat{a} = [\![a]\!]$.
  1. By K2 and K3, $\vdash a = 1;a;1$. As $\vdash 1 = \sum\mathbb{G}$, $\vdash a = \sum\mathbb{G};a;\sum\mathbb{G}$, which by K9 and K10 is equivalent to $\sum\{\alpha;a;\beta \mid \alpha,\beta \in \mathbb{G}\}$. This sum may contain unsatisfiable terms. By corollary 3.3 these are provably equivalent to 0 and hence may be removed by means of K11.
  2. If $\alpha;a;\beta \in [\![a]\!]$ then $R(\alpha;x;\beta) = \{\alpha;a;\beta\} = [\![\alpha;a;\beta]\!]$, by corollary 3.11.
- $\widehat{\phi?} = [\![\phi]\!]$.

1. Clearly $\models \phi? = \sum[\![\phi]\!]$. By theorem 3.2, this equation must be derivable.
2. By lemma 3.10, $[\![\phi]\!]$ only contains guards, so the third condition is easily verified.

- $\widehat{t;u} = \{\sigma;\alpha;\tau \mid \sigma;\alpha \in \widehat{t}, \alpha;\tau \in \widehat{u}\}$.

  1. We need to prove that $\sum \widehat{t;u} = (\sum \widehat{t});(\sum \widehat{u})$ is derivable. Let $A := \widehat{t;u}$ and $B := \{\sigma;\tau \mid \sigma \in \widehat{t}, \tau \in \widehat{u}\}$. By K9 and K10, the above equation reduces to $\sum A = \sum B$. It suffices to show that for every $\sigma \in A$, $\vdash \sigma \leq \sum B$ and vice versa.

     So suppose $\sigma;\alpha;\tau \in A$, with $\sigma;\alpha \in \widehat{t}$ and $\alpha;\tau \in \widehat{u}$. Then $\sigma;\alpha;\alpha;\tau \in B$. Composition is idempotent on tests:

     $$\begin{array}{rcll} \psi?;\psi? & = & (\psi \wedge \psi)? & \text{(T4)} \\ & = & \psi? & \text{(S0)} \end{array}$$

     so $\vdash \alpha;\alpha = \alpha$. Hence $\sigma;\alpha;\tau$ is provably equivalent to a term in B.

     If $\sigma;\alpha;\beta;\tau \in B$ with $\sigma;\alpha \in \widehat{t}$ and $\beta;\tau \in \widehat{u}$ then there are two cases. If $\sigma;\alpha;\beta;\tau$ is not satisfiable, then it is provably equivalent to 0 (use corollary 3.3), hence $\sigma;\alpha;\beta;\tau \leq \sum A$ is derivable. If it *is* satisfiable then α and β are the same, hence $\vdash \sigma;\alpha;\beta;\tau = \sigma;\alpha;\tau$. The latter term is an element of A, so again we have $\vdash \sigma;\alpha;\beta;\tau \leq \sum A$.

  2. Suppose $\tau_1;\alpha;\tau_2 \in \widehat{t;u}$, with $\tau_1;\alpha \in \widehat{t}$ and $\alpha;\tau_2 \in \widehat{u}$. We prove that $R(\tau_1;\alpha;\tau_2) = [\![\tau_1;\alpha;\tau_2]\!]$.

     If $\sigma \in R(\tau_1;\alpha;\tau_2)$ then σ is of the form $\sigma_1;\alpha;\sigma_2$ with $\sigma_i \in R(\tau_i)$. So $\sigma_1;\alpha \in R(\tau_1;\alpha)$ and $\alpha;\sigma_2 \in R(\alpha;\tau_2)$. By the induction hypothesis, $\sigma_1;\alpha \in [\![\tau_1;\alpha]\!]$ and $\alpha;\sigma_2 \in [\![\alpha;\tau_2]\!]$, which implies that $\sigma_1;\alpha;\sigma_2 \in [\![\tau_1;\alpha;\tau_2]\!]$.

     If $\sigma \in [\![\tau_1;\alpha;\tau_2]\!]$ then σ is of the form $\sigma_1;\alpha;\sigma_2$ with $\sigma_1;\alpha \in [\![\tau_1]\!]$ and $\alpha;\sigma_2 \in [\![\tau_2]\!]$. Then $\sigma_1;\alpha \in [\![\tau_1;\alpha]\!] = R(\tau_1;\alpha)$ and $\alpha;\sigma_2 \in [\![\alpha;\tau_2]\!] = R(\alpha;\tau_2)$. Thus $\sigma_i \in R(\tau_i)$ for both i, which implies that $\sigma_1;\alpha;\sigma_2 \in R(\tau_1;\alpha;\tau_2)$.

- $\widehat{t+u} = \widehat{t} \cup \widehat{u}$. The conditions are trivial to check in this case.

- Let T be a set satisfying the first and third conditions of the lemma. That is, T is a finite set of externally guarded Σ-terms such that for every $t \in T$: $R(t) = [\![t]\!]$. We will construct a set U such that these conditions still hold and furthermore: $\vdash \sum U = (\sum T)^*$.

  First of all note that if we remove all guards from T, leaving us with a set T_1 of proper externally guarded terms, then $\vdash (\sum T_1)^* = (\sum T)^*$, as $(t + \psi?)^* = t^*$ is derivable in TC. To see this, note that $(t+1)^* = t^*$ is valid in Kleene algebras (and thus derivable in TC) and that $\psi? \leq 1$ is derivable in TC:

  $$\begin{array}{rcll} \psi? + 1 & = & \psi? + \top? & \text{(T3)} \\ & = & (\psi \vee \top)? & \text{(T3)} \\ & = & \top? & \text{(S0)} \\ & = & 1 & \text{(T3)} \end{array}$$

  So we may assume that T contains no guards.

  If every term in T has the same initial guard α and the same final guard β (such as in the case that T is either empty or a singleton) we distinguish two cases.

  - If $\alpha \neq \beta$ we let U be $\mathbb{G} \cup T$. We need to prove that $(\sum T)^* = 1 + (\sum T)$ is derivable. To see this, note that $\sum T;\sum T = \sum\{t;u \mid t,u \in T\}$ by distributivity. Each term in the latter sum is provably equal to 0 as $\vdash t;u = t;\alpha;\beta;u$ and α and β are incompatible. Hence, reasoning in Kleene algebra:

    $$\begin{array}{rcll} (\sum T)^* & = & 1 + (\sum T) + (\sum T);(\sum T);(\sum T)^* & \text{(K12)} \\ & = & 1 + (\sum T) + 0;(\sum T)^* & \\ & = & 1 + (\sum T) & \text{(K4)} \end{array}$$

– If $\alpha = \beta$, define T_0 as $\{\sigma \mid \alpha;\sigma;\alpha \in T\}$ (so T_0 is T with its outer guards stripped away). Note that $\sum T$ is provably equal to $\alpha;(\sum T_0);\alpha$. Now we can define U as:

$$\mathbb{G} \cup \{\alpha;\sigma;(\alpha;\sum T_0)^*;\alpha \mid \sigma \in T_0\}$$

Using Kleene algebra reasoning and the fact that $\vdash \alpha;\alpha = \alpha$:

$$\begin{array}{rcl}(\sum T)^* & = & 1 + (\sum T);(\sum T)^* \\ & = & 1 + \sum\{\alpha;\sigma;\alpha;(\sum T)^* \mid \sigma \in T_0\} \\ & = & 1 + \sum\{\alpha;\sigma;\alpha;(\alpha;(\sum T_0);\alpha)^* \mid \sigma \in T_0\} \\ & = & 1 + \sum\{\alpha;\sigma;(\alpha;\alpha;(\sum T_0))^*;\alpha \mid \sigma \in T_0\} \\ & = & 1 + \sum\{\alpha;\sigma;(\alpha;(\sum T_0))^*;\alpha \mid \sigma \in T_0\}\end{array}$$

The proof that for any $\sigma \in T_0$:

$$R(\alpha;\sigma;(\alpha;(\sum T_0))^*;\alpha) = [\![\alpha;\sigma;(\alpha;(\sum T_0))^*;\alpha]\!]$$

is left as an exercise. The proof does not differ much from that given for the ;-case.

We now consider the general case where T may have differing initial and final guards. We proceed by induction on the size of T. The cases where T is empty or a singleton have already been dealt with. So suppose that T is of the form $T_1 \cup \{\alpha;\sigma;\beta\}$ where we have already constructed a set U_1 such that $\vdash (\sum T_1)^* = \sum U_1$.

Apply the method of the ;-case (twice) to $(\alpha;\sigma;\beta);(\sum U_1);\alpha$. This gives us a set U_2 such that $\vdash \sum U_2 = \alpha;\sigma;\beta;(\sum U_1);\alpha$. The terms in this set all have the same initial and final guard α. So the method given above applies, giving us a set U_3 such that $\vdash \sum U_3 = (\sum U_2)^*$. Now apply the trick we used for the ;- and the +-case to $(\sum U_1) + (\sum U_1);(\sum U_3);(\alpha;\sigma;\beta);(\sum U_1)$. This yields the desired U.

We reason as follows:

$$\begin{array}{rcl}\sum U & = & (\sum U_1) + (\sum U_1);(\sum U_3);\alpha;\sigma;\beta;(\sum U_1) \\ & = & (\sum T_1)^* + (\sum T_1)^*;(\sum U_2)^*;\alpha;\sigma;\beta;(\sum T_1)^* \\ & = & (\sum T_1)^* + (\sum T_1)^*;[\alpha;\sigma;\beta;(\sum U_1);\alpha]^*;\alpha;\sigma;\beta;(\sum T_1)^* \\ & = & (\sum T_1)^* + (\sum T_1)^*;\alpha;\sigma;\beta;[(\sum U_1);\alpha;\alpha;\sigma;\beta]^*;(\sum T_1)^* \\ & = & (\sum T_1)^* + (\sum T_1)^*;\alpha;\sigma;\beta;((\sum T_1)^*;\alpha;\sigma;\beta)^*;(\sum T_1)^* \\ & = & (\sum T_1 + \alpha;\sigma;\beta)^*\end{array}$$

$\widehat{t^*}$ is then construed by applying the above method to $\widehat{t}$. □

Theorem 3.14 (Completeness of TC) *If t_1 and t_2 are two program terms such that $\mathsf{TA} \models t_1 = t_2$ then $\mathsf{TC} \vdash t_1 = t_2$.*

Proof.
Build the language model $\mathcal{G}$ relative to the tests that occur in t_1 and t_2. By theorem 3.9 $\mathcal{G}$ is a subalgebra of some test algebra. As equations are preserved under subalgebras, $\mathcal{G} \models t_1 = t_2$. In particular, if we interpret the variables in t_1 and t_2 by means of $[\![\_]\!]$ (see page 13), we get that $[\![t_1]\!] = [\![t_2]\!]$.

By theorem 3.13, $\mathsf{TC} \vdash t_i = \sum \widehat{t_i}$ for $i \in \{1,2\}$. Also $R(\sum \widehat{t_1}) = [\![\sum \widehat{t_1}]\!] = [\![t_1]\!] = [\![t_2]\!] = [\![\sum \widehat{t_2}]\!] = R(\sum \widehat{t_1})$, which implies, by Kozen's completeness theorem for KC (see subsection 2.1) that $\sum \widehat{t_1} = \sum \widehat{t_2}$ is valid in Kleene algebras, hence derivable in TC, as TC contains an axiomatization of Kleene algebras. We conclude $\mathsf{TC} \vdash t_1 = t_2$. □

4 Finite axiomatizability

The given axiomatization TC has as one component an axiomatization of Kleene algebras. Redko ([25], see also [6] for a proof in English) proves that no equational axiomatization of Kleene algebra has a finite base. Thus, our axiomatization of TA contains either rules (as in [26, 15]) or equation-schemas (as in [17, 4]). But Kleene algebra terms correspond to test-free PDL-programs. Maybe the addition of tests takes us from a nonfinite axiomatizable theory to a finite axiomatization. This is conveivable: consider the addition of residuals to Kleene algebra in Action Logic ([23]): Kleene algebra is not finitely axiomatizable, but adding two binary operators $\leftarrow$ and $\rightarrow$ to the signature, interpreted on binary relations as follows:

$$\begin{array}{rcl} S \rightarrow R & := & \{(s,t) \mid \forall u.(uSs \rightarrow uRt)\} \\ R \leftarrow S & := & \{(s,t) \mid \forall u.(tSu \rightarrow sRu)\} \end{array}$$

results in a finitely based variety. Will the same transformation occur when moving from dynamic algebra to test algebra? In other words: does adding the test-operator ? to dynamic algebras result in the possibility of a finite axiomatization, a possibility that is not present in the case of dynamic algebra? The answer turns out to be negative: test algebra does not have a finitely based equational axiomatization. The proof is based on a superficial examination of the proof of Conway ([6]) for Kleene algebra.

First we note that in theorem 3.13 only finitely many valid TC-equations were used to prove for any term t that $\vdash t = \sum \widehat{t}$. This finite subtheory we refer to as TC_1. It consists of S0-8, T1-5, K1-12 (everything of KC except the rules K13 and K14) and the following equations:

$$\begin{array}{l} (a+1)^* \leq a^* \\ a;(b;a)^* = (a;b)^*;a \\ (a+b)^* = a^*;(b;a^*)^* \end{array}$$

We will refer to the theory consisting of K1-12 together with the three additional equations above as KC_0: these are precisely the program-equations of TC_1 that only contain Kleene algebra symbols.

Lemma 4.1 *Let $\widehat{(\_)}$ be defined as in theorem 3.13, with respect to some finite set of propositions Γ. Suppose t is a program term that contains only tests from the Fisher-Ladner closure of Γ. Then $\mathsf{TC}_1 \vdash t = \sum \widehat{t}$.*

Proof.
By close inspection of theorem 3.13. □

Lemma 4.2 *If $t = u$ is a valid program equation then there is a valid equation $t' = u'$ of the Kleene algebra signature such that $\mathsf{TC}_1, t' = u' \vdash t = u$.*

Proof.
This is similar to the proof of 3.14. If $t = u$ is a valid program equation of test algebra then $\sum \widehat{t} = \sum \widehat{u}$ is a valid Kleene algebra equation. This latter equation contains tests ψ? but Kleene algebra treats these just like arbitrary relations, so if we uniformly replace the tests in $\sum \widehat{t}$ and $\sum \widehat{u}$ by new variables, we obtain a valid equation $t' = u'$ of the Kleene algebra signature. Clearly $t' = u'$ implies $\sum \widehat{t} = \sum \widehat{u}$. As $\mathsf{TC}_1 \vdash t = \sum \widehat{t}$ and $\mathsf{TC}_1 \vdash u = \sum \widehat{u}$ (by the previous lemma), $\mathsf{TC}_1, t' = u' \vdash t = u$. □

Theorem 4.3 *No finite equational extension of TC_1 completely axiomatizes the valid test algebra equations.*

Proof.
Suppose $T = \mathsf{TC}_1 \cup \{t_1 = u_1, \ldots t_n = u_n\}$ is a complete axiomatization, where we may assume that

the added equations are of the program sort. This is no restriction, because TC_1 is already complete with respect to valid equations of the boolean sort.

Using lemma 4.2 there must be valid Kleene algebra equations $t'_i = u'_i$ such that $T' \cup \{t'_1 = u'_1, \ldots t'_n = u'_n\}$ is also a complete axiomatization. Let $K = \mathsf{KC}_0 \cup \{t'_1 = u'_1, \ldots t'_n = u'_n\}$ be the subtheory of T' that only contains Kleenean operators and variables.

By the result of Redko on nonfinite axiomatizability of Kleene algebra ([25]) there must be an algebra $\mathcal{A}$ of the Kleene signature and a valid equation $t = u$ such that $\mathcal{A} \models K$, but $\mathcal{A} \not\models t = u$. Conway's proof ([6]) gives us a little more: for $a, b \in \mathcal{A}$, $a; b = 0$ only if either $a = 0$ or $b = 0$ (consider the algebra A_p on page 106 of [6]: see also [1] for a clear exposition of Conway's proof).

To complete the proof, we extend $\mathcal{A}$ to a two-sorted algebra $\mathcal{T}$ of the test algebra signature:

- The Kleenean part of $\mathcal{T}$ is simply $\mathcal{A}$.
- The boolean part consists of the two element boolean algebra, whose elements we will denote by $\bot$ and $\top$. Disjunction and negation are interpreted on these as usual.
- The enables operator is defined as follows:

$$\langle a \rangle p := \begin{cases} \bot & \text{if } a = 0 \text{ or } p = \bot. \\ \top & \text{else.} \end{cases}$$

- The test operator ? sends $\bot$ to 0 and $\top$ to 1.

We must prove that $\mathcal{T}$ is a model of T'. It is already a model of K, so what remains to be done is prove that $\mathcal{T}$ satisfies S0-8 and T1-5.

- S0 is satisfied because the boolean part of $\mathcal{T}$ is the prototypical boolean algebra.
- For S4 we need that $0 \neq 1$ in $\mathcal{A}$. If 0 were equal to 1 in $\mathcal{A}$ then $\mathcal{A}$ would be the trivial Kleene algebra, which satisfies any equation, including $t = u$, which we assumed *not* to hold in $\mathcal{A}$.
- For S5 we need that $a + b = 0 \leftrightarrow (a = 0 \wedge b = 0)$ is derivable from KC_0.
- For S6, suppose $\langle a \rangle \langle b \rangle \top = \top \not\leq \bot = \langle a; b \rangle \top$. Then $a; b = 0$ but neither $a = 0$ nor $b = 0$, in contradiction with Conway's construction.
- In connection with S8, note that $a^* = 1 + (a; a^*)$ holds in $\mathcal{A}$, so $\langle a^* \rangle \top = \langle 1 \rangle \top \vee \langle a; a^* \rangle \top$ (use the fact that S6 holds in $\mathcal{T}$). Using S4, we conclude that $\langle a^* \rangle \top = \top$. That S8 holds in $\mathcal{T}$ immediately follows.
- The other equations are trivial to check.

So $\mathcal{T} \models T'$, but it does not satisfy the valid equation $t = u$, in contradiction with the assumption that T' is complete with respect to valid test algebra equations. □

5 Dynamic negation

In this section we explore the possibility of a *homogeneous* or single-sorted, test-algebra. Suppose we wish to define the PDL-programs without reference to complex tests $\phi?$, that is: where ϕ is something other than an atomic proposition. So suppose we have as our language only atomic actions a, the Kleene operators $0, 1, ;, +, *$ and atomic tests $p?$, where p is an atomic proposition. To prove that this restricted system gives us all the programs of PDL, we need to inductively define $\phi?$ in this format. In case ϕ is a disjunction or a conjunction there is no problem, by the validity of T3 and T4:

$$(\phi \vee \psi)? = \phi? + \psi?$$
$$(\phi \wedge \psi)? = \phi?; \psi?$$

But what about negation? We need some unary operator $\sim$ on arbitrary binary relations such that $\sim \phi? = (\neg\phi)?$. Preferably, $\sim$ should be definable within PDL, so that adding it to the language does not give us more definable relations than PDL: we want an equivalence, not an expansion.

An answer is provided by *dynamic negation*, defined as follows:

$$\sim R = \{(s,s) \in S^2 \mid \neg\exists t.sRt\}$$

In words: $\sim R$ is a test that succeeds on those states that are not in the domain of R. It is easy to see that $\sim \phi? = (\neg\phi)?$ should be valid. It also gives us a solution for the diamond case, as $(\langle a\rangle\phi)?$ is equivalent to $\sim\sim (a;\phi?)$. Furthermore, dynamic negation is harmless, as it is definable within PDL by the equation $\sim R = ([R]\bot)?$.

Dynamic negation was first considered in the setting of natural language semantics, where it was proposed as a candidate for natural language negation that has certain nice properties with respect to anaphora linking ([9]). It is also considered in Van Benthem's [2], where it is identified as one of the first-order definable operators that are *safe for bisimulation*, atomic actions, $0, 1, +, ;, p?$ and composites thereof being the others. [11] contains an axiomatization of the valid equalities in the latter language (it also contains an axiomatization of the $+$-free fragment). The present paper started as an attempt to extend this axiomatization to $*$.

Let us recapitulate. A *dynamic negation algebra* is an algebra of signature $\{0, 1, +, ;, *, \sim, (p?)_{p\in\mathsf{P}}\}$ such that the domain is the set of all binary relations on some set S, the Kleene operators are interpreted as they are in relational Kleene algebra, $\sim$ is interpreted as discussed above and each the constant $p?$ is interpreted as some subset of the identity relation. Note that such an algebra is fixed by the choice of S and the interpretation of $p?$ for any $p \in \mathsf{P}$. Let DNA denote the class of all dynamic relation algebras. In this section we will discuss axiomatization of the variety generated byDNA: we will present a calculus (extending an arbitrary calculus of Kleene algebra) that is sound and complete and demonstrate that the variety does not have a finite equational base.

The calculus, DNC (Dynamic Negation Calculus), consists of three parts:

- An axiomatization of Kleene algebra, such as Kozen's.
- Axioms dealing with dynamic negation, from [11]:

N1	$p? = \sim\sim p?$	(test)
N2	$\sim 1 = 0$	(zero definition)
N3	$a = (\sim\sim a); a$	(domain test)
N4	$\sim (a; b); a \leq a; \sim b$	(modus ponens)
N5	$\sim (a + b) = \sim a; \sim b$	(De Morgan)
N6	$\sim a + \sim b = \sim\sim (\sim a + \sim b)$	(test addition)

- An axiom of induction:

$$\text{Ind} \quad \sim (a^*; b) = \sim b; \sim (a^*; \sim b; a; b) \quad \text{(induction)}$$

N4 deserves some attention: why is it called modus ponens? If we define *dynamic implication* $t \Rightarrow u$ as $\sim (t; \sim u)$ then a special instance of N4 is: $(x \Rightarrow y); x \leq x; \sim\sim y$. This says that if $x \Rightarrow y$ succeeds at a node s and we the take an x-step from s to some t then we must be able to proceed from t with a y-step. N4 is thus a dynamic version of the classical static modus ponens.

We present a few examples of reasoning in DNC:

- Identity definition: $\sim 0 = 1$.

$$\begin{array}{lcll} \sim 0 & = & \sim 0; 1 & \\ & = & \sim 0; \sim\sim 1\,; 1 & \text{(N3)} \\ & = & \sim (0 + \sim 1); 1 & \text{(N5)} \\ & = & \sim\sim 1\,; 1 & \\ & = & 1 & \text{(N3)} \end{array}$$

- ; is idempotent and commutative on negated terms:

$$\begin{array}{lcll} \sim a; \sim a & = & \sim (a + a) & \text{(N5)} \\ & = & \sim a & \end{array}$$

$$\begin{array}{lcll} \sim a; \sim b & = & \sim (a + b) & \text{(N5)} \\ & = & \sim (b + a) & \\ & = & \sim b; \sim a & \text{(N5)} \end{array}$$

- Negated terms are subsets of the diagonal:

$$\begin{array}{lcll} \sim a + 1 & = & \sim a + \sim 0 & \text{(identity definition)} \\ & = & \sim\sim (\sim a + \sim 0) & \text{(N6)} \\ & = & \sim (\sim\sim a; \sim\sim 0) & \text{(N5)} \\ & = & \sim (\sim\sim a; 0) & \text{(identity definition, N2)} \\ & = & \sim 0 & \\ & = & 1 & \text{(identity definition)} \end{array}$$

  In other words: $\sim a \leq 1$.

- Negation satisfies a triple negation law:

$$\begin{array}{lcll} \sim a & = & \sim a + \sim a & \\ & = & \sim\sim (\sim a + \sim a) & \text{(N6)} \\ & = & \sim\sim\sim a & \end{array}$$

- Negation is antitone, for if $a \leq b$ then $\sim b = \sim (a + b) = \sim a; \sim b$ by N5. As $\sim b \leq 1$, $\sim a; \sim b \leq \sim a; 1 = \sim a$, so $\sim b \leq \sim a$.

- The ordering $\leq$ may be defined by ; on negated terms: $\sim a \leq \sim b$ iff $\sim a; \sim b = \sim a$.

  From left to right, suppose $\sim a \leq \sim b$. Then:

$$\begin{array}{lcll} \sim a; \sim b & = & \sim a\,; (\sim a + \sim b) & \text{(as } \sim a \leq \sim b\text{)} \\ & = & (\sim a; \sim a) + (\sim a; \sim b) & \\ & = & \sim a + (\sim a; \sim b) & \text{(idempotency of ; on negated terms)} \\ & = & \sim a & \text{(as } \sim a; \sim b \leq \sim a\text{)} \end{array}$$

  The direction from right to left follows directly from the fact that $\sim a; \sim b \leq \sim b$.

- 0 (and thus 1) is definable by means of negation and composition: $\sim a; a = 0$.

$$\begin{array}{lcll} \sim a; a & = & \sim (a; 1); a & \\ & \leq & a; \sim 1 & \text{(N4)} \\ & = & a; 0 & \text{(N2)} \\ & = & 0 & \end{array}$$

- The quasi-equation $\sim a; b = 0 \rightarrow \sim a \leq \sim b$ is derivable. This rule is analogous to the classical negation rule in sequent calculus. To prove derivability, suppose $\sim a; b = 0$. We must then show that $\sim a; \sim b = \sim a$. We only prove $\geq$, as $\leq$ is trivial, by $\sim b \leq 1$.

$$\begin{array}{lcll} \sim a & = & 1; \sim a & \\ & = & \sim 0; \sim a & \text{(identity definition)} \\ & = & \sim (\sim a; b); \sim a & \text{(assumption)} \\ & \leq & \sim a; \sim b & \text{(N4)} \end{array}$$

- The induction axiom Ind is derivable from the others if as our axiomatization we chose Kozen's, which includes the rules K13 and K14. We only prove $\geq$, the other direction being trivial.

  Define $t := \sim b; \sim (a^*; \sim b; a; b))$. We must prove that $t \leq \sim (a^*; b)$. First we prove that $t; a \leq a; t$.

$$\begin{array}{rcll} t; a & = & \sim b; \sim ([\sim b; a; b] + [a; a^*; \sim b; a; b]); a & \\ & = & \sim b; \sim [\sim b; a; b]; \sim [a; a^*; \sim b; a; b]; a & \text{(N5)} \\ & = & \sim [a; a^*; \sim b; a; b]; \sim [\sim b; a; b]; \sim b; a & \text{(commutativity of ; on tests)} \\ & \leq & \sim [a; a^*; \sim b; a; b]; \sim b; a; \sim b & \text{(N4)} \\ & \leq & \sim [a; a^*; \sim b; a; b]; a; \sim b & (\sim b \leq 1) \\ & \leq & a; \sim (a^*; \sim b; a; b); \sim b & \text{(N4)} \\ & = & a; t & \text{(commutativity of ; on tests)} \end{array}$$

  So $a^*; t; a \leq a^*; a; t \leq a^*; t$. By K14, $a^*; t; a^* \leq a^*; t$. It follows that

$$t; a^*; b \leq a^*; t; a^*; b \leq a^*; t; b \leq a^*; \sim b; b = 0$$

  so $t \leq \sim (a^*; b)$.

  Note that this demonstrates that if we simply add the axiomatization of the $\{0, 1, +, ;, \sim\}$-fragment of [11] to Kozen's axiomatization of Kleene algebra we get a complete system for DNA

As stated, there is a clear one-one correspondence between PDL-programs (or test algebra terms of the program sort) and DNA-terms, which we formalize by giving translations both ways. We begin with the translation dna of test algebra terms into dynamic negation algebra terms. The translation is defined simultaneously on terms of the boolean sort and on those of the program sort.

$$\begin{array}{lcl} \mathsf{dna}(\bot) & = & 0 \\ \mathsf{dna}(p) & = & p? \\ \mathsf{dna}(\phi \vee \psi) & = & \mathsf{dna}(\phi) + \mathsf{dna}(\psi) \\ \mathsf{dna}(\neg\phi) & = & \sim \mathsf{dna}(\phi) \\ \mathsf{dna}(\langle\pi\rangle\phi) & = & \sim\sim (\mathsf{dna}(\pi); \mathsf{dna}(\phi)) \\ & & \\ \mathsf{dna}(0) & = & 0 \\ \mathsf{dna}(1) & = & 1 \\ \mathsf{dna}(a) & = & a \\ \mathsf{dna}(\phi?) & = & \mathsf{dna}(\phi) \\ \mathsf{dna}(\pi_1; \pi_2) & = & \mathsf{dna}(\pi_1); \mathsf{dna}(\pi_2) \\ \mathsf{dna}(\pi_1 + \pi_2) & = & \mathsf{dna}(\pi_1) + \mathsf{dna}(\pi_2) \\ \mathsf{dna}(\pi^*) & = & (\mathsf{dna}(\pi))^* \end{array}$$

Lemma 5.1 $\mathsf{TC} \vdash t_1 = t_2$ *implies* $\mathsf{DNC} \vdash \mathsf{dna}(t_1) = \mathsf{dna}(t_2)$*, where the equation* $t_1 = t_2$ *can be of either sort.*

Proof.
The proof transforms any TC-proof of $t_1 = t_2$ into a DNC-proof of $\mathsf{dna}(t_1) = \mathsf{dna}(t_2)$, starting with the axioms and using the induction hypothesis for the rules of equational logic.

The only problem is the substitution rule for boolean variables $t_1 = t_2 / t_1[p := \phi] = t_2[p := \phi]$ for which no correspondent exists in DNC. This is solved by noticing that any DNC-proof $\mathcal{D}$ of $t_1 = t_2$ involving constants $p?$ may be transformed into a proof of $t_1[p? := \sim t] = t_2[p? := \sim t]$ by replacing the constant $p?$ by $\sim t$ throughout the proof. This may not yield a proper DNC-proof, as it may have substituted instances $\sim t = \sim\sim\sim t$ of N1 at its leaves. But we've seen that these are derivable in DNC.

Now to push the induction through the substitution rule. Suppose $\mathsf{DNC} \vdash \mathsf{dna}(t_1) = \mathsf{dna}(t_2)$. We wish to show that $\mathsf{DNC} \vdash \mathsf{dna}(t_1[p := \phi]) = \mathsf{dna}(t_2[p := \phi])$ for any PDL-proposition ϕ. By the above, $\mathsf{DNC} \vdash \mathsf{dna}(t_1)[p? := \mathsf{dna}(\phi)] = \mathsf{dna}(t_2)[p? := \mathsf{dna}(\phi)]$, as $\mathsf{DNC} \vdash \mathsf{dna}(\phi) = \sim\sim \mathsf{dna}(\phi)$ for any

ϕ (by simple induction over ϕ). The proof is completed by noticing that $\mathsf{dna}(t)[p? := \mathsf{dna}(\phi)]$ equals $\mathsf{dna}(t[p := \phi])$ for any test algebra term t. □

Now we present the translation in the other direction. ta takes any dynamic negation algebra term and translates it into a test algebra term of the program sort (a PDL-program).

$$\begin{array}{lcl}
\mathsf{ta}(0) & = & 0 \\
\mathsf{ta}(1) & = & 1 \\
\mathsf{ta}(a) & = & a \\
\mathsf{ta}(p?) & = & p? \\
\mathsf{ta}(\sim t) & = & ([\mathsf{ta}(t)]\bot)? \\
\mathsf{ta}(t_1; t_2) & = & \mathsf{ta}(t_1); \mathsf{ta}(t_2) \\
\mathsf{ta}(t_1 + t_2) & = & \mathsf{ta}(t_1) + \mathsf{ta}(t_2) \\
\mathsf{ta}(t^*) & = & \mathsf{ta}(t)^*
\end{array}$$

Lemma 5.2 $\mathsf{DNA} \models t_1 = t_2$ *implies* $\mathsf{TA} \models \mathsf{ta}(t_1) = \mathsf{ta}(t_2)$.

Proof.
Let $\mathcal{T}$ be the test algebra over some set S and let σ be some assignment to variables. So σ assigns binary relations on S to program variables (i.e. elements of A) and subsets of S to proposition variables (elements of P). There is a corresponding dynamic negation algebra $\mathcal{D}$ over S by fixing the interpretation of constants $p?$ to $\{(s,s) \mid s \in \sigma(p)\}$. It is easy to prove that for any dynamic negation algebra term t: $[\![t]\!]^{\mathcal{D}}_{\sigma\restriction\mathsf{A}} = [\![\mathsf{ta}(t)]\!]^{\mathcal{T}}_{\sigma}$ (in words: t is interpreted as the same relation in $\mathcal{D}$ under the assignment σ restricted to the program variables A as $\mathsf{ta}(t)$ is interpreted in $\mathcal{T}$ under σ).

The lemma now follows immediately. For suppose $\mathsf{DNA} \models t_1 = t_2$. Consider any test algebra $\mathcal{T}$ and any assignment σ. Let $\mathcal{D}$ be the corresponding dynamic negation algebra. Then $[\![\mathsf{ta}(t_1)]\!]^{\mathcal{T}}_{\sigma} = [\![t_1]\!]^{\mathcal{D}}_{\sigma\restriction\mathsf{A}} = [\![t_2]\!]^{\mathcal{D}}_{\sigma\restriction\mathsf{A}}$ (as $t_1 = t_2$ is valid) $= [\![\mathsf{ta}(t_2)]\!]^{\mathcal{T}}_{\sigma}$. Thus $\mathsf{TA} \models \mathsf{ta}(t_1) = \mathsf{ta}(t_2)$. □

Theorem 5.3 DNC *is sound and complete with respect to* DNA.

Proof.
Soundness is trivial to verify. For completeness, suppose $\mathsf{DNA} \models t_1 = t_2$. By lemma 5.2 $\mathsf{TA} \models \mathsf{ta}(t_1) = \mathsf{ta}(t_2)$. TC is complete with respect to TA (theorem 3.14) so $\mathsf{TC} \vdash \mathsf{ta}(t_1) = \mathsf{ta}(t_2)$. Using lemma 5.1 we get that $\mathsf{DNC} \vdash \mathsf{dna}(\mathsf{ta}(t_1)) = \mathsf{dna}(\mathsf{ta}(t_2))$. By induction on dynamic negation algebra terms t, $\mathsf{DNC} \vdash t = \mathsf{dna}(\mathsf{ta}(t))$, which we may use to conclude $\mathsf{DNC} \vdash t_1 = t_2$. □

The proof that test algebra is not finitely equationally definable also carries over to the dynamic negation algebra setting:

Theorem 5.4 *There is no finite set of sound* DNA*-equations* D *such that* $D \vdash t_1 = t_2$ *for any* DNA*-valid equation* $t_1 = t_2$.

Proof.
Suppose there is such a set D. Let T be the test algebra theory consisting of $\mathsf{ta}[D]$ (test algebra translations of the D-equations), and of S0, S6, T1 and T3. Then T must completely axiomatize test algebra. The proof of this consists of two parts:

- For any test algebra term ϕ of the boolean sort: $T \vdash \phi? = \mathsf{ta}(\mathsf{dna}(\phi))$ (in fact, only S0, S6, T1 and T3 are needed). For any test algebra term π of the program sort: $T \vdash \pi = \mathsf{ta}(\mathsf{dna}(\pi))$ (again, we only need S0, S6, T1 and T3). The proof of this is by simultaneous induction on test algebra terms of both sorts.
- Suppose $\mathsf{TA} \models t_1 = t_2$. Then by lemma 5.1 and completeness (theorems 3.14 and 5.3) $\mathsf{DNA} \models \mathsf{dna}(t_1) = \mathsf{dna}(t_2)$. By the assumption $D \vdash \mathsf{dna}(t_1) = \mathsf{dna}(t_2)$. We may now translate the proof of this to the test algebra setting to deduce that $\mathsf{ta}[D] \vdash \mathsf{ta}(\mathsf{dna}(t_1)) = \mathsf{ta}(\mathsf{dna}(t_2))$. Using the fact that T derives $\mathsf{ta}(\mathsf{dna}(t_i)) = t_i$, we deduce $T \vdash t_1 = t_2$.

As T is finite this is in contradiction with theorem 4.3. □

6 Further research

We have proved that extending any axiomatization of Kleene algebra with the equations S0-8 and T1-5 gives us a complete axiomatization of test algebras. In other words: test algebra is finitely equationally axiomatizable *relative to* Kleene algebra. We have also demonstrated that this relativeness cannot be dropped: no finite set of equations axiomatizes test algebra. Finally, we have proved similar theorems for a single-sorted version of test algebra.

A first topic for further research concerns making a similar move as Pratt made with his Action Logic ([23]): Kleene algebra is not finitely axiomatizable, but adding residuals (see page 18) to the language *makes* it so. The question is what happens to test algebra if we add such operations to the language.

Adding relation-complement to the language almost gives us RAT (relation algebra with transitive closure, see [19]), or equivalently Peirce algebra ([5]) with $*$. It *almost* gives us RAT because converse is present in RAT. It is then obvious what extension next to consider: test algebra with converse. This is an important extension to consider, as converse was present in the original PDL-paper [21], although it disappeared in many later versions of PDL. An axiomatization of test algebra would depend heavily on the axiomatization of PDL with converse supplied by [20].

Adding relation complement to the language is perhaps too strong an option, gives rise to programs that do not seem to be based on real-live programs: the program $-a$ (the complement of a) gives as output any state such that it is impossible to output this state when executing the program a on the input state. This is in contradiction with our intuition that programs should in some sense *construct* the output from the input. From this perspective it makes more sense to add just program intersection to the language (the corresponding logic is called IPDL in [10]).

Finally, we could impose a more functional structure on programs. This could be done by adding a construct that removes all pairs (s, t) from a relation such that R is not a function on s. An alternative is to consider test algebras where the domain consists of all partial functions on a set S. This would mean dropping $+$ from the language as this doesn't preserve functionality. There is a connection to DPDL (deterministic PDL, see [10] for a description and references), but it is not obvious, as the latter logic does have sum in its repertoire.

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) for encouraging and aiding me in this research, and actually reading unreadable earlier versions of this paper.

References

[1] L. Aceto, W. Fokkink, and A. Ingólfsdóttir. On a question of Salomaa: the equational theory of regular expressions over a singleton alphabet is not finitely axiomatizable. To appear in the BRICS Report Series, 1996.

[2] J.F.A.K. van Benthem. Programming operations that are safe for bisimulation. CSLI Report 93-179, Center for the Study of Language and Information, Stanford University, 1993. To appear in *Studia Logica*.

[3] J.F.A.K. van Benthem. *Exploring logical dynamics*. CSLI Publications and FoLLI, Stanford University, 1996.

[4] S.L. Bloom and Z. Ésik. Equational axioms for regular sets. *Mathematical structures in computer science*, 3:1–24, 1993.

[5] C. Brink, K. Britz, and R. Schmidt. Peirce algebras. *Formal Aspects of Computing*, 6:1–20, 1994.

[6] J.H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

[7] M.J. Fisher and R.E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, 1979.

[8] R. Goldblatt. *Logics of Time and Computation. Second Edition*, volume 7 of *CSLI Lecture Notes*. CSLI Publications, 1992.

[9] J. Groenendijk and M. Stokhof. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100, 1991.

[10] D. Harel. Dynamic logic. In D.M. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic, Vol. II*, pages 497–604. Reidel, Dordrecht, 1984.

[11] M.J. Hollenberg. An axiomatization of strong negation and relational composition. Logic Group Preprint Series 151, Dept. of Philosophy, Utrecht University, 1995. To appear in the *Journal of Logic, Language and Information*,
`ftp://ftp.phil.ruu.nl/pub/logic/PREPRINTS/preprint151.ps.Z`.

[12] 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.

[13] 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.

[14] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:234–241, 1983.

[15] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110:2:366–390, May 1994.
`http://www.cs.cornell.edu/Info/People/kozen/papers/ka.ps`.

[16] D. Kozen and F. Smith. Kleene algebra with tests: completeness and decidability.
`http://www.cs.cornell.edu/Info/People/kozen/papers/kat.ps`, 1996.

[17] D. Krob. A complete system of B-rational identities. *Theoretical Computer Science*, 89(2):207–343, October 1991.

[18] I. Németi. Dynamic algebras of programs. In *Proceedings Fundamentals of Computation Theory*, volume 117 of *LNCS*, pages 281–290. Springer-Verlag, 1981.

[19] K.C. Ng. *Relation Algebras with Transitive Closure*. PhD thesis, University of California, Berkely, 1984.

[20] R. Parikh. The completeness of propositonal dynamic logic. In *Proceedings 7th Symposium on Mathematical Foundations of Computer Science*, volume 64 of *LNCS*, pages 403–415. Springer-Verlag, 1978.

[21] V. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.

[22] 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.

[23] V. Pratt. Action logic and pure induction. In *Logics in AI: European Workshop JELIA '90*, volume 478 of *LNCS*, pages 97–120. Springer-Verlag, September 1990.

[24] V. Pratt. Dynamic algebras: examples, constructions, applications. *Studia Logica*, 50:571–605, 1991.

[25] V.N. Redko. On defining relations for the algebra of regular events. *Ukrainskii Matematicheskii Zhurnal*, 16:120–126, 1964. In Russian.

[26] A. Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of the ACM*, 13(1):158–169, January 1966.

[27] 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.

[28] R.S. Streett and E.A. Emerson. An automata theoretic procedure for the propositional mu-calculus. *Information and Computation*, 81:249–264, 1989.

[29] V. Trnková and J. Reiterman. Dynamic algebra with test. *Journal of Computer and System Sciences*, 35:229–242, 1987.