Kleene Algebra with Dynamic Tests

Igor Sedlár

Institute of Computer Science of the Czech Academy of Sciences

LLAMA Seminar
15. 11. 2023
Motivation: Reasoning about programs

Reasoning about...

- program equivalence  \(\rightarrow\) shorter code
- halting / divergence  \(\rightarrow\) code runs
- correctness  \(\rightarrow\) code does what it should
Motivation: Reasoning about programs

Reasoning about...
- program equivalence → shorter code
- halting / divergence → code runs
- correctness → code does what it should

...programs...

\[ E, F \ := \ a \in \Sigma \mid \text{skip} \mid \text{abort} \mid E; F \mid \text{if } B \text{ then } E \text{ else } F \mid \text{while } B \text{ do } F \]

where \( B \) is a Boolean formula over a set of variables \( \Pi \).
Motivation: Reasoning about programs

Reasoning about...
- program equivalence → shorter code
- halting / divergence → code runs
- correctness → code does what it should

...programs...

\[ E, F := a \in \Sigma \mid \text{skip} \mid \text{abort} \mid E; F \mid \text{if } B \text{ then } E \text{ else } F \mid \text{while } B \text{ do } F \]

where \( B \) is a Boolean formula over a set of variables \( \Pi \).

...using “logics of programs”.

Igor Sedlár (ICS CAS)
Outline

1. Kleene algebra with tests (KAT)
2. KA with dynamic tests (aKAT) and fragments (KA with domain)
3. aKAT and PDL $\rightarrow$ relational aKAT is EXPTIME-complete
4. Language completeness for aKAT $\rightarrow$ aKAT = language aKAT
5. Relational completeness for aKAT $\rightarrow$ relational aKAT = language aKAT = aKAT $\rightarrow$ aKAT is EXPTIME complete
6. Completeness and complexity for fragments $\rightarrow$ completeness for all $\rightarrow$ aKA is EXPTIME-complete

See arxiv.org/abs/2311.06937
1. Kleene algebra with tests
Kleene algebra with tests (KAT) – 1

(Cohet et al. 1996; Kozen 1996, 1997; Kozen and Smith, 1997)

Syntax (KA + tests)

\[ e, f := a \in \Sigma \mid 0 \mid 1 \mid e + f \mid e \cdot f \mid e^* \mid p \in \Pi \mid p^\perp \]
Kleene algebra with tests (KAT) – 1

(Cohet et al. 1996; Kozen 1996, 1997; Kozen and Smith, 1997)

Syntax (KA + tests)

\[ e, f := a \in \Sigma \mid 0 \mid 1 \mid e + f \mid e \cdot f \mid e^* \mid p \in \Pi \mid p^\perp \]

Axioms

\begin{align*}
(e \cdot f) \cdot g & \equiv e \cdot (f \cdot g) \quad (1) & 1 + (e \cdot e^*) & \leq e^* \quad (8) \\
e \cdot 1 & \equiv e \equiv 1 \cdot e \quad (2) & 1 + (e^* \cdot e) & \leq e^* \quad (9) \\
(e + f) + g & \equiv e + (f + g) \quad (3) & f + (e \cdot g) & \leq g \implies e^* \cdot f \leq g \quad (10) \\
0 + e & \equiv 0 \equiv e + 0 \quad (4) & f + (g \cdot e) & \leq g \implies f \cdot e^* \leq g \quad (11) \\
e & \equiv e + e \quad (5) \\
e \cdot (f + g) & \equiv (e \cdot f) + (e \cdot g) \quad (6) \\
(e + f) \cdot g & \equiv (e \cdot g) + (f \cdot g) \quad (7)
\end{align*}

\[ \uparrow \text{idempotent semirings} \]

where \( e \leq f \) means \( e + f \equiv f \) and \( b, c \in \Lambda = \Pi \cup \Pi^\perp \).

(Kappé 2022, 2023)
Kleene algebra with tests (KAT) – 2

Encoding programs...

- Boolean formulas: DFN over \( \Lambda \)
  
  e.g. \( \neg(p \leftrightarrow \neg q) \) is \( p \cdot q + p^\perp \cdot q^\perp \) if \( \Pi = \{p, q\} \)

- \( E; F \) is \( E \cdot F \)

- skip as 1 and abort as 0

- if \( B \) then \( E \) else \( F \) is \( (B \cdot E) + (\neg B \cdot F) \)

- while \( B \) do \( E \) is \( (B \cdot E)^* \cdot (\neg B) \)

...and their properties

- equivalence: \( e \equiv f \)

- divergence \( e \equiv 0 \), halting
  
  \( e \not\equiv 0 \)

- correctness
  
  \( (B \cdot e) \cdot (\neg C) \equiv 0 \)
Kleene algebra with tests (KAT) – 2

Encoding programs...

- Boolean formulas: DFN over \( \Lambda \)
  
  \[ E; F \text{ is } E \cdot F \]

- skip as 1 and abort as 0

- if \( B \) then \( E \) else \( F \) is \( (B \cdot E) + (\neg B \cdot F) \)

- while \( B \) do \( E \) is \( (B \cdot E)^* \cdot (\neg B) \)

...and their properties

- equivalence: \( e \equiv f \)

- divergence \( e \equiv 0 \), halting \( e \neq 0 \)

- correctness
  
  \( (B \cdot e) \cdot (\neg C) \equiv 0 \)

Properties of KAT:

- Sound and complete for relational models and language models based on guarded strings

- Eq. theory PSPACE-complete

- Quasi-eq. theory \( \Sigma_1^0 \)-complete; fragment with assumptions \( e \equiv 0 \) reduces to eq. th.
2. Kleene algebra with dynamic tests
Kleene algebra with dynamic tests – 1

Syntax:

\[ e, f := a \in \Sigma \mid p \in \Pi \mid 0 \mid 1 \mid e + f \mid e \cdot f \mid e^* \mid e^\perp \mid e^\top \]

Read \( e^\perp \) as \textit{“e diverges”} and \( e^\top \) as \textit{“e halts”}. 
Kleene algebra with dynamic tests – 1

Syntax:

\[ e, f := a \in \Sigma \mid p \in \Pi \mid 0 \mid 1 \mid e + f \mid e \cdot f \mid e^{*} \mid e^{\perp} \mid e^{\top} \]

Read \( e^{\perp} \) as “\( e \) diverges” and \( e^{\top} \) as “\( e \) halts”.

→ \( \perp \) is antidomain and \( \top \) is domain of KA with domain (Desharnais et al. 2006, 2011)
Kleene algebra with dynamic tests – 1

Syntax:

\[ e, f := a \in \Sigma \mid p \in \Pi \mid 0 \mid 1 \mid e + f \mid e \cdot f \mid e^* \mid e^\bot \mid e^\top \]

Read \( e^\bot \) as “\( e \) diverges” and \( e^\top \) as “\( e \) halts”.

\( \rightarrow \) \( \bot \) is antidomain and \( \top \) is domain of KA with domain (Desharnais et al. 2006, 2011)

Axioms: KA +

\[ e^\bot \cdot e \equiv 0 \quad (12) \]
\[ (e \cdot f)^\bot \equiv (e \cdot f^\top)^\bot \quad (13) \]
\[ e^\bot + e^\top \equiv 1 \quad (14) \]
\[ p^\top \equiv p \quad (15) \]
\[ e^\top \equiv e^\bot^\bot \quad (16) \]

This is “antidomain KAT”, or aKAT.
Let $\mathbb{E}$ be the set of all expressions.

**Definition 1**

The set $\mathbb{F}$ of **formulas** (over $\Sigma$ and $\Pi$) is defined by the following grammar:

$$\phi, \psi ::= p \in \Pi \mid 0 \mid 1 \mid \phi + \psi \mid \phi \cdot \psi \mid e \bot \mid e \top,$$

where $e \in \mathbb{E}$.

**Definition 2**

An expression $e \in \mathbb{E}$ is **testable** iff it does not contain occurrences of $\top$. A **test** is an expression $e \top$ where $e$ is testable. A **parameter** is either a test or an element of $\Pi$.

Notation: $\Phi \bot = \{ \phi \bot \mid \phi \in \Phi \}$ and $\Phi \pm = \Phi \cup \Phi \bot$.
<table>
<thead>
<tr>
<th>K</th>
<th>Base</th>
<th>$\top$ applies to</th>
<th>$\bot$ applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{KA}$</td>
<td>$\Sigma$</td>
<td>nothing</td>
<td>nothing</td>
</tr>
<tr>
<td>$\text{KA}(\Phi)$</td>
<td>$\Sigma \cup \Phi$</td>
<td>nothing</td>
<td>nothing</td>
</tr>
<tr>
<td>$\text{dKA}$</td>
<td>$\Sigma$</td>
<td>everything</td>
<td>noting</td>
</tr>
<tr>
<td>$\text{aKA}$</td>
<td>$\Sigma$</td>
<td>everything</td>
<td>everything</td>
</tr>
<tr>
<td>$\text{KAT}$</td>
<td>$\Sigma \cup \Pi$</td>
<td>nothing</td>
<td>$\Pi$</td>
</tr>
<tr>
<td>$\text{KAT}(\Phi)$</td>
<td>$\Sigma \cup \Phi$</td>
<td>nothing</td>
<td>$\Phi$</td>
</tr>
<tr>
<td>$\text{dKAT}$</td>
<td>$\Sigma \cup \Pi$</td>
<td>everything</td>
<td>$\Pi$</td>
</tr>
<tr>
<td>$\text{aKAT}$</td>
<td>$\Sigma \cup \Pi$</td>
<td>everything</td>
<td>everything</td>
</tr>
</tbody>
</table>

$\Phi \subseteq \mathcal{E}$ is a set of parameters. $\mathcal{RE}_{\text{KA}(\Phi)}$ ($\mathcal{RE}_{\text{KAT}(\Phi)}$) is the set of regular expressions over $\Phi$ ($\Phi^\pm$); notation: $\mathcal{RE}(\Phi)$ ($\mathcal{RE}(\Phi^\pm)$).
We write \( e \equiv^K f \) if \( e \equiv f \) and \( e, f \in \mathbb{E}_K \).
Definition 3

A relational model (for $\Sigma$ and $\Pi$) is a structure $M = \langle X, \text{rel}_M, \text{sat}_M \rangle$ where

- $X$ is a set
- $\text{rel}_M : \Sigma \rightarrow 2^{X \times X}$
- $\text{sat}_M : \Pi \rightarrow 2^X$.

Intuition: $\langle x, y \rangle \in \text{rel}_M(a)$, or

$$x \xrightarrow{a} y,$$

if action $a$ may lead from state $x$ to state $y$; $x \in \text{sat}_M(p)$, or

$$(M, x) \models p,$$

if proposition $p$ is satisfied in state $x$.

**Note:** This is a Kripke frame for $\Sigma$ as the “basis” of the modal signature.
Relational models – 2

Definition 4

For each relational model $M = \langle X, \text{rel}_M, \text{sat}_M \rangle$ (for $\Sigma$ and $\Pi$) we define the $M$-interpretation of $E$ as the function $\llbracket - \rrbracket_M : E \rightarrow 2^{X \times X}$ such that:

\[
\begin{align*}
\llbracket a \rrbracket_M &= \text{rel}_M(a) \\
\llbracket p \rrbracket_M &= 1_{\text{sat}_M(p)} \\
\llbracket 0 \rrbracket_M &= \emptyset \\
\llbracket 1 \rrbracket_M &= X
\end{align*}
\]

\[
\begin{align*}
\llbracket e + f \rrbracket_M &= \llbracket e \rrbracket_M \cup \llbracket f \rrbracket_M \\
\llbracket e \cdot f \rrbracket_M &= \llbracket e \rrbracket_M \circ \llbracket f \rrbracket_M \\
\llbracket e^* \rrbracket_M &= (\llbracket e \rrbracket_M)^* = \bigcup_{n \geq 0} \llbracket e \rrbracket^n_M \\
\llbracket e^\perp \rrbracket_M &= a(\llbracket e \rrbracket_M) = \bigcup_{R \subseteq 1_X} (R \circ \llbracket e \rrbracket_M = \emptyset) \\
\llbracket e^\top \rrbracket_M &= d(\llbracket e \rrbracket_M) = X \setminus a(\llbracket e \rrbracket_M)
\end{align*}
\]

where, for $R \subseteq 2^{X \times X}$, $R^0 = 1_X$ and $R^{n+1} = R^n \circ R$. Expressions $e$ and $f$ are relationally equivalent) iff $\llbracket e \rrbracket_M = \llbracket f \rrbracket_M$ for all $M$. 

Igor Sedlár (ICS CAS)
Relational models – 3

Note that

\[
[e^\perp]_M = \{ \langle x, x \rangle \mid \neg \exists y : \langle x, y \rangle \in [e]_M \}
\]

\[
[e^\top]_M = \{ \langle x, x \rangle \mid \exists y : \langle x, y \rangle \in [e]_M \}
\]

In other words, \( \perp \) is dynamic negation of Dynamic Predicate Logic (Groenendijk and Stokhof, 1991) and \( \top \) is dynamic double negation.
Note that

\[\llbracket e^\perp \rrbracket_M = \{ \langle x, x \rangle \mid \neg \exists y : \langle x, y \rangle \in \llbracket e \rrbracket_M \}\]

\[\llbracket e^\top \rrbracket_M = \{ \langle x, x \rangle \mid \exists y : \langle x, y \rangle \in \llbracket e \rrbracket_M \}\]

In other words, \(\perp\) is dynamic negation of Dynamic Predicate Logic (Groenendijk and Stokhof, 1991) and \(\top\) is dynamic double negation.

If \(\llbracket e \rrbracket_M\) is the start-halt relation for (program) \(e\), then

- \(\llbracket e^\perp \rrbracket_M\) represents the set of states where \(e\) diverges
- \(\llbracket e^\top \rrbracket_M\) represents the set of states where \(e\) halts
3. aKAT and PDL
Satisfaction of formulas – 1

It is easy to prove by induction on the structure of $\phi$ that $\llbracket \phi \rrbracket_M \subseteq 1_X$ for all formulas $\phi$ and all $M$. We will write

$$(M, x) \models \phi \iff \langle x, x \rangle \in \llbracket \phi \rrbracket_M$$
Satisfaction of formulas – 1

It is easy to prove by induction on the structure of $\phi$ that $[[\phi]]_M \subseteq 1_X$ for all formulas $\phi$ and all $M$. We will write

$$(M, x) \models \phi \iff \langle x, x \rangle \in [[\phi]]_M$$

Observation 1

For all $\phi$ and all $M$:

1. $(M, x) \models \phi$ iff $\langle x, x \rangle \in [[\phi^\top]]_M$
2. $(M, x) \not\models 0$ and $(M, x) \models 1$ for all $x$
3. $(M, x) \models \phi^\perp$ iff $(M, x) \not\models \phi$
4. $(M, x) \models \phi + \psi$ iff $(M, x) \models \phi$ or $(M, x) \models \psi$
5. $(M, x) \models \phi \cdot \psi$ iff $(M, x) \models \phi$ and $(M, x) \models \psi$
6. $(M, x) \models (e \cdot \phi)^\perp$ iff there is $y$ such that $\langle x, y \rangle \in [[e]]_M$ and $(M, y) \models \phi$
7. $(M, x) \models (e \cdot \phi^\perp)^\perp$ iff for all $y$, $\langle x, y \rangle \in [[e]]_M$ implies that $(M, y) \models \phi$
Satisfaction of formulas – 2

\[
\langle e \rangle \phi \\
\langle x, x \rangle \in \llbracket (e \cdot \phi) \llbracket_\perp \rrbracket_M \iff \neg \exists y : \langle x, y \rangle \in \llbracket (e \cdot \phi) \llbracket_\perp \rrbracket_M
\\
\iff \langle x, x \rangle \notin \llbracket (e \cdot \phi) \llbracket_\perp \rrbracket_M
\\
\iff \exists y : \langle x, y \rangle \in \llbracket e \cdot \phi \rrbracket_M
\\
\iff \exists y : \langle x, y \rangle \in \llbracket e \rrbracket_M \land \langle y, y \rangle \in \llbracket \phi \rrbracket_M
\]

\[
\langle x, x \rangle \in \llbracket (e \cdot \phi^\perp) \llbracket_\perp \rrbracket_M \iff \neg \exists y : \langle x, y \rangle \in \llbracket (e \cdot \phi^\perp) \llbracket_\perp \rrbracket_M
\\
\iff \forall y, z : \langle x, y \rangle \in \llbracket e \rrbracket_M \implies \langle y, z \rangle \notin \llbracket \phi^\perp \rrbracket
\\
\iff \forall y : \langle x, y \rangle \in \llbracket e \rrbracket_M \implies \langle y, y \rangle \in \llbracket \phi \rrbracket_M
\]
aKAT = "PDL in disguise"

We obtain the usual semantics of (programs and formulas) of PDL within aKAT. PDL programs form a specific fragment $E_{PDL}$ of $E$. aKAT is "one sorted test algebra".

Theorem 1
The problem of deciding relational equivalence between arbitrary expressions in $E$ is $\text{EXPTIME}$-complete.

Proof.
Lower bound: the ($\text{EXPTIME}$-hard) membership problem for polynomial-space alternating Turing machines reduces to the problem of satisfiability of PDL-formulas in relational models: For each machine $A$ and input $t$, there is a PDL-formula $F_{A,t}$ such that $A$ accepts $t$ iff $F_{A,t}$ is satisfiable.

Upper bound: for every $e \in E$ there is a poly. computable equivalent $e' \in E_{PDL}$.

Question:
Do we need the full aKAT language for arbitrary $F_{A,t}$?
We obtain the usual semantics of (programs and formulas) of PDL within aKAT. PDL programs form a specific fragment $E_{PDL}$ of $E$. aKAT is “one sorted test algebra”.

**Theorem 1**

*The problem of deciding relational equivalence between arbitrary expressions in $E$ is EXPTIME-complete.*

*Proof.* Lower bound: the (EXPTIME-hard) membership problem for polynomial-space alternating Turing machines reduces to the problem of satisfiability of PDL-formulas in relational models: For each machine $A$ and input $t$, there is a PDL-formula $F_{A,t}$ such that $A$ accepts $t$ iff $F_{A,t}$ is satisfiable.

Upper bound: for every $e \in E$ there is a poly. computable equivalent $e' \in E_{PDL}$.

**Question:** Do we need the full aKAT language for arbitrary $F_{A,t}$?
4. Language completeness
Disclaimer: We draw heavily on Hollenberg’s (1998) relational completeness proof for an equational axiomatization of Test Algebras (relationally valid equations between PDL-programs), itself combining modal logic with (Kozen and Smith, 1997). We simplify and generalize.

Let $\Phi$ be a finite set of parameters $\phi_1, \ldots, \phi_n$. An atom over $\Phi$ is a sequence $\psi_1 \ldots \psi_n$ where $\psi_i \in \{\phi_i, \phi_i^\perp\}$. Notation $G \triangleleft \phi$ means “$\phi$ appears in atom $G$”. $A(\Phi)$ is the set of all atoms over $\Phi$.

A guarded string over $\Phi$ is any sequence of the form

$$G_1 a_1 G_2 \ldots a_{n-1} G_n$$

where each $G_i \in A(\Phi)$ and $a_j \in \Sigma$. $GS(\Phi)$ is the set of all guarded strings over $\Phi$. 

Guarded strings – 2

**Fusion product** is a partial binary operation on $\mathbb{GS}(\Phi)$ defined as follows:

$$xG \odot Hy = \begin{cases} xy & G = H \\ \text{undefined} & G \neq H \end{cases}$$

Fusion product is lifted to $\Phi$-guarded languages $K, L \subseteq \mathbb{GS}(\Phi)$ as expected:

$$L \odot K = \{ w \odot u \mid w \in L \land u \in K \}.$$
Definition 5

The algebra of \( \Phi \)-guarded languages is

\[
\text{GL}(\Phi) = \langle 2^{\text{GS}(\Phi)}, 2^\text{A}(\Phi), \cup, \Diamond, *, \perp, \top, \emptyset, \text{A}(\Phi) \rangle
\]

where \( K^* = \bigcup_{n \geq 0} K^n \) (\( K^0 = \text{A}(\Phi) \) and \( K^{n+1} = K^n \Diamond K \)) and

\[
L^{\perp} = \{ G \in \text{A}(\Phi) \mid \{ G \} \Diamond L = \emptyset \} \quad \text{and} \quad L^{\top} = \{ G \in \text{A}(\Phi) \mid \{ G \} \Diamond L \neq \emptyset \}.
\]

Definition 6

If \( \Phi \) is a finite set of parameters, then the standard \( \Phi \)-interpretation of \( \text{RE}(\Phi^{\pm}) \) is the unique homomorphism \( [-]_\Phi : \text{RE}(\Phi^{\pm}) \to \text{GL}(\Phi) \) such that

\[
[a]_\Phi = \{ GaH \mid G, H \in \text{A}(\Phi) \} \quad \text{and} \quad [\phi]_\Phi = \{ G \mid G \in \text{A}(\Phi) \& G \triangleright \phi \}
\]

for \( a \in \Sigma \) and \( \phi \in \Phi^{\pm} \).
Theorem 2 (Essentially (Kozen and Smith 1997))

Let $\Phi$ be a finite set of parameters. For all $e, f \in E_{\text{KAT}}(\Phi)$,

$$
e \equiv_{\text{KAT}(\Phi)} f \iff [e]_\Phi = [f]_\Phi$$

Consequently, we have

$$[e]_\Gamma = [f]_\Gamma \implies e \equiv f$$

for all $e, f \in \mathbb{RE}(\Gamma^\perp) \subseteq E$.

The converse fails! $a^T a^\perp \top \in A(\Gamma)$ for $\Gamma = \{a^T, a^\perp \top\}$. Hence, $[a^T \cdot a^\perp]_\Gamma \neq \emptyset$. → Pay attention to consistency of atoms! Moreover, $[a^\perp \cdot a]_\Gamma = \{GaH \mid G \triangleleft a^\perp\} \neq \emptyset$. → Consider only consistent one-step gstrings!
We don’t distinguish between a non-empty sequence of expressions \( e_1 \ldots e_n \) and the expression \( e_1 \cdot \ldots \cdot e_n \) (assuming some fixed bracketing). An atom \( G \) is consistent iff \( G \not\equiv 0 \). \( \mathbb{C}(\Phi) \) is the set of all consistent atoms over \( \Phi \).

A consistently guarded string over \( \Phi \) is any guarded string \( G_1 a_1 \ldots a_{n-1} G_n \) where all \( G_i \in \mathbb{C}(\Phi) \). \( \mathbb{CS}(\Phi) \) is the set of consistently guarded strings over \( \Phi \).

**Definition 7**

*The algebra of consistently \( \Phi \)-guarded languages is*

\[
\mathbb{CL}(\Phi) = \langle 2^{\mathbb{CS}(\Phi)}, 2^{\mathbb{C}(\Phi)}, \cup, \Diamond, *, \bot, \top, \emptyset, \mathbb{C}(\Phi) \rangle
\]

where \( K^* = \bigcup_{n \geq 0} K^n \), \( K^0 = \mathbb{C}(\Phi) \) and \( K^{n+1} = K^n \Diamond K \) and

\[
L_{\bot} = \{ G \in \mathbb{C}(\Phi) \mid \{ G \} \Diamond L = \emptyset \} \quad L^\top = \{ G \in \mathbb{C}(\Phi) \mid \{ G \} \Diamond L \neq \emptyset \}.
\]

Not a subalgebra of \( \mathbb{GL}(\Phi) \) although of course \( \mathbb{CS}(\Phi) \subseteq \mathbb{GS}(\Phi) \).
Definition 8

Let $\Gamma$ be a set of parameters. The canonical $\Gamma$-interpretation of $E$ is the unique homomorphism $[-] : E \rightarrow CL(\Gamma)$ such that

$$[a]_\Gamma = \{ GaH \in CS(\Gamma) \mid GaH \not\equiv 0 \} \quad [p]_\Gamma = \{ G \mid G \in C(\Gamma) \& G \leq p \}$$

for all $a \in \Sigma$ and $p \in \Pi$.

Lemma 1 (Language soundness)

Let $\Gamma$ be any finite set of parameters. For all $e, f \in E$:

$$e \equiv f \implies [e]_\Gamma = [f]_\Gamma.$$
Language completeness – 1

**Definition 9**

We define \( \hat{\cdot} \) as the smallest function \( \mathbb{RE}(\Gamma) \rightarrow \mathbb{RE}(\Gamma^\pm) \) such that (for \( \phi \in \Gamma \) and \( a \in \Sigma \))

\[
\hat{\phi} = \sum \{G \in \mathbb{C}(\Gamma) \mid G \leq \phi\} \quad \hat{a} = \sum [a]_\Gamma \quad \hat{1} = \sum \mathbb{C}(\Gamma)
\]

and that commutes with 0, ·, + and *.

**Lemma 2**

For all \( e \in \mathbb{RE}(\Gamma) \), \( e \equiv \hat{e} \).

**Lemma 3**

If \( \Gamma \) is “FL-closed”, then \([e]_\Gamma = [\hat{e}]_\Gamma \) for all \( e \in \mathbb{RE}(\Gamma) \).
Theorem 3 (Language completeness)

Let $E \subseteq E$ be finite and let $\Gamma$ be the FL-closure of the sets of tests of subformulas of elements of $E$. Then, for all $e, f \in E$:

$$e \equiv f \iff \lceil e \rceil_\Gamma = \lceil f \rceil_\Gamma$$

Proof.

The implication from left to right follows from Lemma 1. The converse implication is established as follows:

$$\begin{align*}
\lceil e \rceil_\Gamma &= \lceil f \rceil_\Gamma & \text{Lemma 3} \\
\hat{e} \Gamma &= \hat{f} \Gamma & \text{Theorem 2} \\
\hat{e} \equiv & \hat{f} & \text{by def.} \\
& \text{Lemma 2} \\
e & \equiv f
\end{align*}$$
5. Relational completeness
**Definition 10**

*We define the function* $\text{cay} : 2^{\text{CS}(\Gamma)} \rightarrow 2^{\text{CS}(\Gamma) \times \text{CS}(\Gamma)}$ *as follows:*

$$\text{cay}(L) = \{ \langle w, w \diamond u \rangle \mid w \in \text{CS}(\Gamma) \land u \in L \}$$

**Definition 11**

*Define the relational model* $\text{CS}(\Gamma) = \langle \text{CS}(\Gamma), \text{rel}_{\text{CS}(\Gamma)}, \text{sat}_{\text{CS}(\Gamma)} \rangle$ *where*

$$\text{rel}_{\text{CS}(\Gamma)}(a) = \text{cay}(\llbracket a \rrbracket_{\Gamma}) \quad \text{sat}_{\text{CS}(\Gamma)}(p) = \{ w \mid \text{last}(w) \leq p \}$$

*for* $a \in \Sigma$ *and* $p \in \Pi$. 
Lemma 4

If $\Gamma$ is FL-closed, then for all $e \in \mathbb{RE}(\Gamma)$,

$$\text{cay} ([e]_{\Gamma}) = [e]_{CS(\Gamma)}.$$

Theorem 4 (Relational completeness)

For all $e, f \in \mathbb{E}$:

$$e \equiv f \iff (\forall M)([e]_M = [f]_M)$$

Theorem 5

The problem of deciding equivalence of arbitrary expressions is EXPTIME-complete.
6. Completeness and complexity of fragments
Completeness and complexity of fragments – 1

Theorem 6

Take a finite $E \subseteq E$ and $e, f \in E$. The following are equivalent:

1. $e \equiv^K f$
2. $[[e]]_\Gamma = [[f]]_\Gamma$ where $\Gamma$ is the FL-closure of $\text{St}(E)$
3. $[[e]]_M = [[f]]_M$ for all relational models $M$
Completeness and complexity of fragments – 2

Lemma 5

Let $e'$ be the result of replacing every occurrence of $a_n$ in $e$ by an occurrence of $a_{2n}$ and replacing every occurrence of $p_n$ by an occurrence of $(a_{2n+1})^\top$. Then

$$e \equiv f \iff e' \equiv f'.$$

Proof.
Left to right: Equivalence is preserved under substitution. Moreover, clearly $p' \equiv (p')^\top$.

Right to left: If $e \not\equiv f$, then there is a relational model $M$ where $J e K_M \neq J f K_M$ (Theorem 4). We define $M'$ by taking the universe $X$ of $M$ and stipulating that $\text{rel}_{M'}(a_m) = \{ \text{rel}_M(a_n) | m = 2n \}$ and $\text{sat}_{M'}(p_n) = \emptyset$. It can be shown by induction on $g$ that $J g K_M = J g' K_{M'}$. Only the base case is interesting. The base case for $a_n$: $J (a_n) K_{M'} = J (a_{2n}) K_{M'} = J a_n K_M$. The base case for $p_n$: $J (p_n) K_{M'} = J (a_{2n+1})^\top K_{M'} = \{ \langle x, x \rangle | \exists y. \langle x, y \rangle \in J a_{2n+1} K_{M'} \} = J p_n K_M$. Now clearly $J e' K_{M'} \neq J f' K_{M'}$ and so $e' \not\equiv f'$ by relational soundness.
Completeness and complexity of fragments – 2

Lemma 5

Let \( e' \) be the result of replacing every occurrence of \( a_n \) in \( e \) by an occurrence of \( a_{2n} \) and replacing every occurrence of \( p_n \) by an occurrence of \( (a_{2n+1})^\top \). Then

\[
e \equiv f \iff e' \equiv f'.
\]

Proof. Left to right: Equivalence is preserved under substitution. Moreover, clearly \( p' \equiv (p')^\top \).

Right to left: If \( e \not\equiv f \), then there is a relational model \( M \) where \( \llbracket e \rrbracket_M \neq \llbracket f \rrbracket_M \) (Theorem 4). We define \( M' \) by taking the universe \( X \) of \( M \) and stipulating that

\[
\text{rel}_{M'}(a_m) = \begin{cases} 
\text{rel}_M(a_n) & m = 2n \\
1_{\text{sat}_M(p_n)} & m = 2n + 1
\end{cases} \quad \text{sat}_{M'}(p) = \emptyset
\]

It can be shown by induction on \( g \) that \( \llbracket g \rrbracket_M = \llbracket g' \rrbracket_{M'} \). Only the base case is interesting. The base case for \( a_n \):

\[
\llbracket (a_n)' \rrbracket_{M'} = \llbracket a_{2n} \rrbracket_{M'} = \llbracket a_n \rrbracket_M.
\]

The base case for \( p_n \):

\[
\llbracket (p_n)' \rrbracket_{M'} = \llbracket (a_{2m+1})^\top \rrbracket_{M'} = \{ \langle x, x \rangle \mid \exists y. \langle x, y \rangle \in \llbracket a_{2n+1} \rrbracket_{M'} \} = \llbracket p_n \rrbracket_M.
\]

Now clearly

\[
\llbracket e' \rrbracket_{M'} \neq \llbracket f' \rrbracket_{M'} \quad \text{and so} \quad e' \neq f' \quad \text{by relational soundness}.
\]

Igor Sedlár (ICS CAS)  Kleene Algebra with Dynamic Tests  LLAMA, 15. 11. 2023  26/30
Completeness and complexity of fragments – 3

Theorem 7

The problem of deciding equivalence between aKA-expressions is EXPTIME-complete.

Proof. The problem is in EXPTIME since so is deciding equivalence between arbitrary expressions. The problem is EXPTIME-complete thanks to Lemma 5: deciding equivalence between arbitrary expressions can be polynomially reduced to deciding equivalence between aKA-expressions. The former is EXPTIME-complete by Theorem 1.

aKA is the “Kleene algebra with domain” of Desharnais and Struth (2011).
Completeness and complexity of fragments – 4

PSPACE-complete  EXPTIME-complete
7. Conclusion
Conclusion

We discussed various extensions of KA(T) with $\top$ (domain) and $\bot$ (antidomain). This family contains KA, KAT, PDL (aKAT), and various versions of “Kleene algebra with domain” that appeared in the literature.

Results:

- aKAT and all fragments considered are sound and complete with respect to relational and (parametrized) guarded-language models
- aKAT and aKAT are EXPTIME-complete
Conclusion

Problems:

1. Are dKA and dKAT EXPTIME-hard?

2. Do fragments of quasi-eq. theories of aKAT and its fragments with assumptions $e \equiv 0$ reduce to their eq. theories?

3. Are there natural fragments of aKAT that are stronger than KAT but still have a PSPACE-complete eq. theory?

4. What are the natural automata-theoretic formulation of the various versions of Kleene algebra with dynamic tests considered here?

Thank you!


