

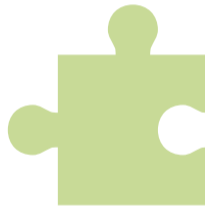
# Kleene Algebra with Observations

*Tobias Kappé* Paul Brunet Jurriaan Rot

Alexandra Silva Jana Wagemaker Fabio Zanasi

PPLV Research Trip, Galway







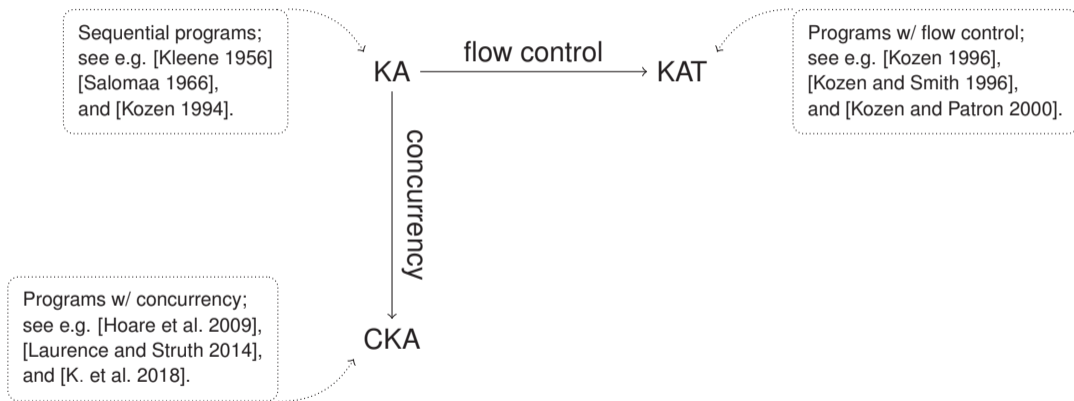
Sequential programs;  
see e.g. [Kleene 1956]  
[Salomaa 1966],  
and [Kozen 1994].

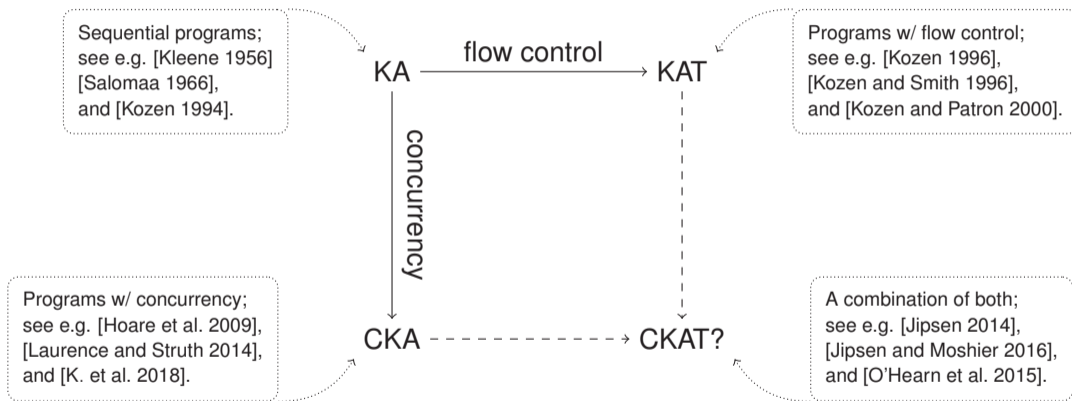
KA

Sequential programs;  
see e.g. [Kleene 1956]  
[Salomaa 1966],  
and [Kozen 1994].

KA  $\xrightarrow{\text{flow control}}$  KAT

Programs w/ flow control;  
see e.g. [Kozen 1996],  
[Kozen and Smith 1996],  
and [Kozen and Patron 2000].







$$e, f ::= p \mid 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e^* \mid e \parallel f$$
$$p, q ::= \perp \mid \top \mid t \in \mathcal{T} \mid p \vee q \mid p \wedge q \mid \bar{p}$$

$$e, f ::= p \mid 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e^* \mid e \parallel f$$

KAT terms

$$p, q ::= \perp \mid \top \mid t \in T \mid p \vee q \mid p \wedge q \mid \bar{p}$$

CKA terms

$$e, f ::= p \mid 0 \mid 1 \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e^* \mid e \parallel f$$

$$p, q ::= \perp \mid \top \mid t \in T \mid p \vee q \mid p \wedge q \mid \bar{p}$$

$$e + 0 \equiv e \quad e + e \equiv e \quad e + f \equiv f + e \quad e + (f + g) \equiv (e + f) + g$$

$$e \cdot 1 \equiv e \quad e \equiv 1 \cdot e \quad e \cdot 0 \equiv 0 \quad 0 \equiv 0 \cdot e \quad e \cdot (f \cdot g) \equiv (e \cdot f) \cdot g$$

$$e \cdot (f + g) \equiv e \cdot f + e \cdot g \quad 1 + e \cdot e^* \equiv e^* \quad e + f \cdot g \leq g \implies f^* \cdot e \leq g$$

$$(e + f) \cdot g \equiv e \cdot g + f \cdot g \quad 1 + e^* \cdot e \equiv e^* \quad e + f \cdot g \leq f \implies e \cdot g^* \leq f$$

$$e \leq f \iff e + f \equiv f$$

$$p \vee \perp \equiv p \quad p \vee q \equiv q \vee p \quad p \vee \bar{p} \equiv \top \quad p \vee (q \vee r) \equiv (p \vee q) \vee r$$

$$p \wedge \top \equiv p \quad p \wedge q \equiv q \wedge p \quad p \wedge \bar{p} \equiv \perp \quad p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$$

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r) \quad p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$$

$$p \vee q \equiv p + q \quad p \wedge q \equiv p \cdot q \quad \top \equiv 1 \quad \perp \equiv 0$$

$$\begin{array}{llll} e \parallel f \equiv f \parallel e & e \parallel 1 \equiv e & e \parallel 0 \equiv 0 & (e + f) \parallel g \equiv e \parallel g + f \parallel g \\ e \parallel (f \parallel g) \equiv (e \parallel f) \parallel g & & & (e \parallel f) \cdot (g \parallel h) \leq (e \cdot g) \parallel (f \cdot h) \end{array}$$

$$\begin{aligned} e \parallel f &\equiv f \parallel e & e \parallel 1 &\equiv e & e \parallel 0 &\equiv 0 & (e + f) \parallel g &\equiv e \parallel g + f \parallel g \\ e \parallel (f \parallel g) &\equiv (e \parallel f) \parallel g & (e \parallel f) \cdot (g \parallel h) &\leq (e \cdot g) \parallel (f \cdot h) \end{aligned}$$

In particular:  $e \cdot f \leq e \parallel f$ .

$$p \cdot e \cdot \bar{p} \leq (p \parallel e) \cdot \bar{p}$$



$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \end{aligned}$$

$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \end{aligned}$$

$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \\ &\equiv (p \cdot \bar{p}) \parallel e \end{aligned}$$

$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \\ &\equiv (p \cdot \bar{p}) \parallel e \\ &\equiv (p \wedge \bar{p}) \parallel e \end{aligned}$$

$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \\ &\equiv (p \cdot \bar{p}) \parallel e \\ &\equiv (p \wedge \bar{p}) \parallel e \\ &\equiv \perp \parallel e \end{aligned}$$

$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \\ &\equiv (p \cdot \bar{p}) \parallel e \\ &\equiv (p \wedge \bar{p}) \parallel e \\ &\equiv \perp \parallel e \\ &\equiv 0 \parallel e \end{aligned}$$

$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \\ &\equiv (p \cdot \bar{p}) \parallel e \\ &\equiv (p \wedge \bar{p}) \parallel e \\ &\equiv \perp \parallel e \\ &\equiv 0 \parallel e \\ &\equiv 0 \end{aligned}$$

$$\begin{aligned} p \cdot e \cdot \bar{p} &\leq (p \parallel e) \cdot \bar{p} \\ &\equiv (p \parallel e) \cdot (\bar{p} \parallel 1) \\ &\leq (p \cdot \bar{p}) \parallel (e \cdot 1) \\ &\equiv (p \cdot \bar{p}) \parallel e \\ &\equiv (p \wedge \bar{p}) \parallel e \\ &\equiv \perp \parallel e \\ &\equiv 0 \parallel e \\ &\equiv 0 \end{aligned}$$







Drop  $e \parallel 0 \equiv 0$ ?



Drop  $e \parallel 0 \equiv 0$ ?



Drop exchange law?

Non-congruence?

Drop  $e \parallel 0 \equiv 0$ ?



Drop exchange law?



See also [Bergstra and Ponse 2011].



See also [Bergstra and Ponse 2011].



See also [Bergstra and Ponse 2011].



See also [Bergstra and Ponse 2011].



$$p \wedge q \equiv p \cdot q \rightsquigarrow p \wedge q \leq p \cdot q$$

$$~~1 \equiv \top~~$$

$$p \wedge q \equiv p \cdot q \rightsquigarrow p \wedge q \leq p \cdot q$$

$$~~1 \equiv \top~~$$

Our contribution: a study of the *sequential fragment* (i.e., without  $\parallel$ ).

$$p \wedge q \equiv p \cdot q \rightsquigarrow p \wedge q \leq p \cdot q$$

$$~~1 \equiv \top~~$$

Our contribution: a study of the *sequential fragment* (i.e., without  $\parallel$ ).

- A semantics  $\llbracket - \rrbracket$  that sends a term to its *observation language*.

$$p \wedge q \equiv p \cdot q \rightsquigarrow p \wedge q \leq p \cdot q$$

$$1 \equiv \top$$

Our contribution: a study of the *sequential fragment* (i.e., without  $\parallel$ ).

- A semantics  $\llbracket - \rrbracket$  that sends a term to its *observation language*.
- Practically feasible decision procedure w.r.t. semantics.

$$p \wedge q \equiv p \cdot q \rightsquigarrow p \wedge q \leq p \cdot q$$

$$~~1 \equiv \top~~$$

Our contribution: a study of the *sequential fragment* (i.e., without  $\parallel$ ).

- A semantics  $\llbracket - \rrbracket$  that sends a term to its *observation language*.
- Practically feasible decision procedure w.r.t. semantics.
- Soundness & completeness:  $e \equiv f$  if and only if  $\llbracket e \rrbracket = \llbracket f \rrbracket$ .

$$p \wedge q \equiv p \cdot q \rightsquigarrow p \wedge q \leq p \cdot q$$

$$1 \equiv \top$$

Our contribution: a study of the *sequential fragment* (i.e., without  $\parallel$ ).

- A semantics  $\llbracket - \rrbracket$  that sends a term to its *observation language*.
- Practically feasible decision procedure w.r.t. semantics.
- Soundness & completeness:  $e \equiv f$  if and only if  $\llbracket e \rrbracket = \llbracket f \rrbracket$ .

These form the foundation for extending to the full syntax.

$$\llbracket 0 \rrbracket = \emptyset$$

$$\llbracket a \rrbracket = \{a\}$$

$$\llbracket e + f \rrbracket = \llbracket e \rrbracket \cup \llbracket f \rrbracket$$

$$\llbracket e^* \rrbracket = \llbracket e \rrbracket^*$$

$$\llbracket 1 \rrbracket = \{\epsilon\}$$

$$\llbracket p \rrbracket = \{\alpha \in \text{At} : \alpha \leq p\}$$

$$\llbracket e \cdot f \rrbracket = \llbracket e \rrbracket \cdot \llbracket f \rrbracket ,$$

$$\begin{array}{llll} \llbracket 0 \rrbracket = \emptyset & \llbracket a \rrbracket = \{a\} & \llbracket e + f \rrbracket = \llbracket e \rrbracket \cup \llbracket f \rrbracket & \llbracket e^* \rrbracket = \llbracket e \rrbracket^* \\ \llbracket 1 \rrbracket = \{\epsilon\} & \llbracket p \rrbracket = \{\alpha \in \text{At} : \alpha \leq p\} & \llbracket e \cdot f \rrbracket = \llbracket e \rrbracket \cdot \llbracket f \rrbracket, & \end{array}$$

$$\alpha \equiv \alpha \wedge \alpha \leq \alpha \cdot \alpha \quad \text{but} \quad \llbracket \alpha \rrbracket \not\subseteq \llbracket \alpha \cdot \alpha \rrbracket$$



$$\frac{w, x \in (\Sigma \cup \text{At})^* \quad \alpha \in \text{At}}{w\alpha x \preceq w\alpha\alpha x}$$

$$\frac{w, x \in (\Sigma \cup \text{At})^* \quad \alpha \in \text{At}}{w\alpha x \preceq w\alpha\alpha x}$$

Let  $\llbracket - \rrbracket^{\preceq}$  be the  $\preceq$ -downclosure of  $\llbracket - \rrbracket$ ; then

### Lemma (Soundness)

*If  $e \equiv f$ , then  $\llbracket e \rrbracket^{\preceq} = \llbracket f \rrbracket^{\preceq}$ .*

Can we decide whether  $\llbracket e \rrbracket^{\lambda} = \llbracket f \rrbracket^{\lambda}$ ?

Can we decide whether  $\llbracket e \rrbracket^\lambda = \llbracket f \rrbracket^\lambda$ ?

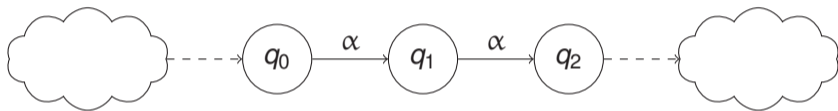
- 1 Translate  $e$  to automaton  $A_e$  such that  $L(A_e) = \llbracket e \rrbracket^\lambda$ .

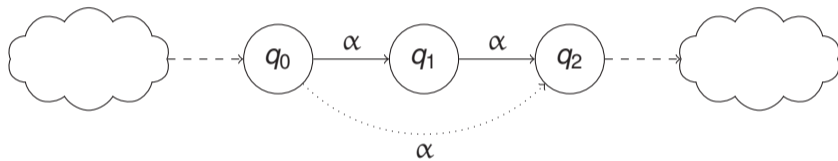
Can we decide whether  $\llbracket e \rrbracket^\lambda = \llbracket f \rrbracket^\lambda$ ?

- 1 Translate  $e$  to automaton  $A_e$  such that  $L(A_e) = \llbracket e \rrbracket^\lambda$ .
- 2 Translate  $f$  to automaton  $A_f$  such that  $L(A_f) = \llbracket f \rrbracket^\lambda$ .

Can we decide whether  $\llbracket e \rrbracket^{\lambda} = \llbracket f \rrbracket^{\lambda}$ ?

- 1 Translate  $e$  to automaton  $A_e$  such that  $L(A_e) = \llbracket e \rrbracket^{\lambda}$ .
- 2 Translate  $f$  to automaton  $A_f$  such that  $L(A_f) = \llbracket f \rrbracket^{\lambda}$ .
- 3 Decide whether  $L(A_e) = L(A_f)$ .







$$\epsilon(0) = 0$$

$$\epsilon(1) = 1$$

$$\epsilon(a) = 0$$

$$\epsilon(p) = 0$$

$$\epsilon(e + f) = \epsilon(e) \vee \epsilon(f)$$

$$\epsilon(e \cdot f) = \epsilon(e) \wedge \epsilon(f)$$

$$\epsilon(e^*) = 1$$

See also [Brzozowski 1964].

$$\delta(0, a) = \delta(1, a) = \emptyset$$

$$\delta(a, a') = \{1 : a = a'\}$$

$$\delta(p, a) = \emptyset$$

$$\delta(e + f, a) = \delta(e, a) \cup \delta(f, a)$$

$$\delta(e \cdot f, a) = \{e' \cdot f : e' \in \delta(e, a)\} \cup \Delta(e, f, a)$$

$$\delta(e^*, a) = \{e' \cdot e^* : e' \in \delta(e, a)\},$$

$$\Delta(e, f, a) = \begin{cases} \delta(f, a) & \epsilon(e) = 1 \\ \emptyset & \epsilon(e) = 0 \end{cases}$$

See also [Antimirov 1996].

$$\zeta(0, \alpha) = \delta(1, \alpha) = \emptyset$$

$$\zeta(a, \alpha) = \emptyset$$

$$\zeta(p, \alpha) = \{1 : \alpha \leq p\}$$

$$\zeta(e + f, \alpha) = \zeta(e, \alpha) \cup \zeta(f, \alpha)$$

$$\zeta(e \cdot f, \alpha) = \{e' \cdot f : e' \in \zeta(e, \alpha)\} \cup Z(e, f, \alpha)$$

$$\zeta(e^*, \alpha) = \{e' \cdot e^* : e' \in \zeta(e, \alpha)\},$$

$$Z(e, f, \alpha) = \begin{cases} \zeta(f, \alpha) & \epsilon(e') = 1 \text{ for some } e' \in \zeta(e, \alpha) \cup \{e\} \\ \emptyset & \text{otherwise} \end{cases}$$

Let  $\rho(e)$  be the expressions reached through  $\delta$  or  $\zeta$  from  $e$ .

## Lemma

*For any  $e$ , we have that  $A_e = \langle \rho(e), \delta \cup \zeta, \epsilon \rangle$  is a finite non-deterministic automaton.*

Let  $\rho(e)$  be the expressions reached through  $\delta$  or  $\zeta$  from  $e$ .

## Lemma

*For any  $e$ , we have that  $A_e = \langle \rho(e), \delta \cup \zeta, \epsilon \rangle$  is a finite non-deterministic automaton.*

## Lemma

*For any  $e$ , we have that  $L(A_e) = \llbracket e \rrbracket^{\zeta}$ .*

Let  $\rho(e)$  be the expressions reached through  $\delta$  or  $\zeta$  from  $e$ .

## Lemma

*For any  $e$ , we have that  $A_e = \langle \rho(e), \delta \cup \zeta, \epsilon \rangle$  is a finite non-deterministic automaton.*

## Lemma

*For any  $e$ , we have that  $L(A_e) = \llbracket e \rrbracket^{\lambda}$ .*

## Theorem

*Given  $e, f$ , we can decide whether  $\llbracket e \rrbracket^{\lambda} = \llbracket f \rrbracket^{\lambda}$ .*

An expression  $e$  is ...

An expression  $e$  is ...

- ... *atomic* if its Boolean subterms are atoms.



An expression  $e$  is ...

- ... *atomic* if its Boolean subterms are atoms.
- ... *closed* if  $\llbracket e \rrbracket = \llbracket e \rrbracket^{\sphericalangle}$ .

An expression  $e$  is ...

- ... *atomic* if its Boolean subterms are atoms.
- ... *closed* if  $\llbracket e \rrbracket = \llbracket e \rrbracket^{\neg}$ .

## Lemma

*Let  $e$  and  $f$  be atomic and closed. If  $\llbracket e \rrbracket^{\neg} = \llbracket f \rrbracket^{\neg}$ , then  $e \equiv f$ .*

## Lemma

*For every  $e$ , we can construct an atomic and closed  $\hat{e}$  such that  $e \equiv \hat{e}$ .*

## Lemma

*For every  $e$ , we can construct an atomic and closed  $\hat{e}$  such that  $e \equiv \hat{e}$ .*

## Proof sketch.

Idea: convert the automaton for  $e$  back into an expression.

## Lemma

*For every  $e$ , we can construct an atomic and closed  $\hat{e}$  such that  $e \equiv \hat{e}$ .*

## Proof sketch.

Idea: convert the automaton for  $e$  back into an expression.

Formally, this is the least  $x_e$  that solves the system generated by:

$$\epsilon(e) + \sum_{e' \in \delta(e, a)} a \cdot x_{e'} + \sum_{e' \in \zeta(e, \alpha)} \alpha \cdot x_{e'} \leq x_e$$



## Theorem

*If  $\llbracket e \rrbracket^{\surd} = \llbracket f \rrbracket^{\surd}$ , then  $e \equiv f$ .*

## Theorem

*If  $\llbracket e \rrbracket^{\surd} = \llbracket f \rrbracket^{\surd}$ , then  $e \equiv f$ .*

## Proof.

First, note that

$$\llbracket \hat{e} \rrbracket = \llbracket \hat{e} \rrbracket^{\surd} = \llbracket e \rrbracket^{\surd} = \llbracket f \rrbracket^{\surd} = \llbracket \hat{f} \rrbracket^{\surd} = \llbracket \hat{f} \rrbracket$$

## Theorem

*If  $\llbracket e \rrbracket^{\surd} = \llbracket f \rrbracket^{\surd}$ , then  $e \equiv f$ .*

## Proof.

First, note that

$$\llbracket \hat{e} \rrbracket = \llbracket \hat{e} \rrbracket^{\surd} = \llbracket e \rrbracket^{\surd} = \llbracket f \rrbracket^{\surd} = \llbracket \hat{f} \rrbracket^{\surd} = \llbracket \hat{f} \rrbracket$$

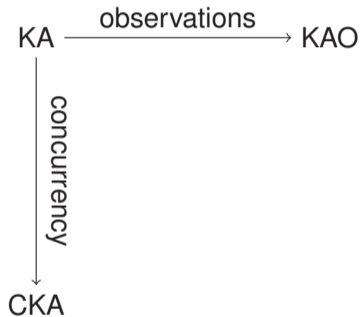
Since  $\hat{e}$  and  $\hat{f}$  are atomic and closed, we conclude

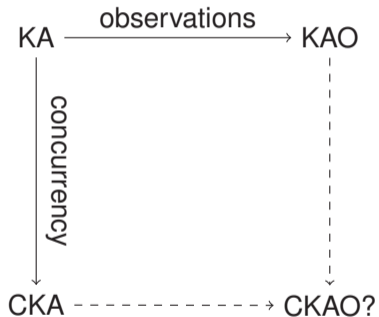
$$e \equiv \hat{e} \equiv \hat{f} \equiv f$$











- 1 Build programming language (e.g. NetKAT) on top of this.

- 1 Build programming language (e.g. NetKAT) on top of this.
- 2 Isolation: some form of persistence of observations?

- 1 Build programming language (e.g. NetKAT) on top of this.
- 2 Isolation: some form of persistence of observations?
- 3 Optimise decision procedure; exploit transitive property?

- 1 Build programming language (e.g. NetKAT) on top of this.
- 2 Isolation: some form of persistence of observations?
- 3 Optimise decision procedure; exploit transitive property?
- 4 Possible applications in weak memory.

Thank you for your attention

The logo for GoNeCo, where the letters 'GoNeCo' are rendered in a large, black, serif font. The letter 'o' is replaced by a black silhouette of a cat. The letter 'N' has an orange cat silhouette perched on its top bar. The letter 'e' has a black silhouette of a cat's head inside its loop. The letter 'o' at the end has a black silhouette of a cat's head inside its loop.






<https://coneco-project.org>






Extended paper: <https://arxiv.org/abs/1811.10401>.






Icons (CC-BY) from The Noun Project (<https://thenounproject.com>)

- "Puzzle" by BlueTip design.
- "Scales" by jai.
- "The Scream" by Jonah Bethlehem.
- "Crossroads Decision" by Anna Sophie.
- "Biking" by tulpahn.
- "Chicken" by BomSymbols.



-  V. M. Antimirov [1996]. “Partial Derivatives of Regular Expressions and Finite Automaton Constructions”. In: *Theor. Comput. Sci.* 155.2, pp. 291–319. DOI: 10.1016/0304-3975(95)00182-4.
-  J. A. Bergstra and A. Ponse [2011]. “Proposition algebra”. In: *ACM Trans. Comput. Log.* 12.3, 21:1–21:36. DOI: 10.1145/1929954.1929958.
-  J. A. Brzozowski [1964]. “Derivatives of Regular Expressions”. In: *J. ACM* 11.4, pp. 481–494. DOI: 10.1145/321239.321249.
-  T. Hoare et al. [2009]. “Concurrent Kleene Algebra”. In: *CONCUR*, pp. 399–414. DOI: 10.1007/978-3-642-04081-8\_27.
-  P. Jipsen [2014]. “Concurrent Kleene Algebra with Tests”. In: *RAMiCS*, pp. 37–48. DOI: 10.1007/978-3-319-06251-8\_3.

-  P. Jipsen and M. A. Moshier [2016]. “Concurrent Kleene algebra with tests and branching automata”. In: *J. Log. Algebr. Meth. Program.* 85.4, pp. 637–652. DOI: [10.1016/j.jlamp.2015.12.005](https://doi.org/10.1016/j.jlamp.2015.12.005).
-  T. Kappé et al. [2018]. “Concurrent Kleene Algebra: Free Model and Completeness”. In: *ESOP*, pp. 856–882. DOI: [10.1007/978-3-319-89884-1\\_30](https://doi.org/10.1007/978-3-319-89884-1_30).
-  S. C. Kleene [1956]. “Representation of Events in Nerve Nets and Finite Automata”. In: *Automata Studies*, pp. 3–41.
-  D. Kozen [1994]. “A Completeness Theorem for Kleene Algebras and the Algebra of Regular Events”. In: *Inf. Comput.* 110.2, pp. 366–390. DOI: [10.1006/inco.1994.1037](https://doi.org/10.1006/inco.1994.1037).
-  — [1996]. “Kleene algebra with tests and commutativity conditions”. In: *TACAS*, pp. 14–33. DOI: [10.1007/3-540-61042-1\\_35](https://doi.org/10.1007/3-540-61042-1_35).

-  D. Kozen and M. Patron [2000]. “Certification of Compiler Optimizations Using Kleene Algebra with Tests”. In: *CL*, pp. 568–582. DOI: 10.1007/3-540-44957-4\_38.
-  D. Kozen and F. Smith [1996]. “Kleene Algebra with Tests: Completeness and Decidability”. In: *CSL*, pp. 244–259. DOI: 10.1007/3-540-63172-0\_43.
-  M. R. Laurence and G. Struth [2014]. “Completeness Theorems for Bi-Kleene Algebras and Series-Parallel Rational Pomset Languages”. In: *RAMiCS*, pp. 65–82. DOI: 10.1007/978-3-319-06251-8\_5.
-  P. W. O’Hearn et al. [2015]. “On the relation between Concurrent Separation Logic and Concurrent Kleene Algebra”. In: *J. Log. Algebr. Meth. Program.* 84.3, pp. 285–302. DOI: 10.1016/j.jlamp.2014.08.002.
-  A. Salomaa [1966]. “Two Complete Axiom Systems for the Algebra of Regular Events”. In: *J. ACM* 13.1, pp. 158–169. DOI: 10.1145/321312.321326.