A Full experimental results

Figure 6 contains the full experimental results, which were described in Section 6 and summarized in Figure 4.

B Proof of Theorem 7.3

We need the following lemma.

Lemma B.1. *If $X, Z \in RE$ are clean and normalized then $\delta^+(XZ) = \delta^+(X)Z \cup \delta^+(Z)$; if $X = S*$ then $\delta^+(X) = \delta^+(S)X$.*

Proof. We prove by induction over X that

$$\delta^+(XZ) = \delta^+(X)Z \cup \delta^+(Z).$$

It follows from working with normalized regexes that in a concatenation node the first element is not a concatenation and we apply case analysis over the first element, that is not an intersection or complement because here we only consider standard regexes.

Base case $X = \varepsilon$**.** Follows immediately because $\delta(\varepsilon) = \emptyset$.

Induction case $X = \psi Y$**.**
Then $\delta(XZ) = \mathbf{IF}(\psi, \varepsilon, \bot)\cdot YZ = \mathbf{IF}(\psi, YZ, \bot)$, so $\mathbf{Q}(\delta(XZ)) = \{YZ\}$ and thus

$$\begin{array}{rcl}\delta^+(XZ) & = & \{YZ\} \cup \delta^+(YZ)\\ & \overset{IH}{=} & \{YZ\} \cup \delta^+(Y)Z \cup \delta^+(Z)\\ & = & (\{Y\} \cup \delta^+(Y))Z \cup \delta^+(Z)\\ & = & \delta^+(X)Z \cup \delta^+(Z)\end{array}$$

Induction case $X = (X_1|X_2)Y$**.**
Then $\delta(XZ) = (\delta(X_1YZ)\,|\,\delta(X_2YZ))$, so

$$\begin{array}{rcl}\delta^+(XZ) & = & \delta^+(X_1YZ) \cup \delta^+(X_2YZ)\\ & \overset{2\times IH}{=} & \delta^+(X_1Y)Z \cup \delta^+(X_2Y)Z \cup \delta^+(Z)\\ & \overset{2\times IH}{=} & (\delta^+(X_1)Y \cup \delta^+(Y))Z \cup\\ & & (\delta^+(X_2)Y \cup \delta^+(Y))Z \cup \delta^+(Z)\\ & = & (\delta^+(X_1)Y \cup \delta^+(X_2)Y \cup \delta^+(Y))Z \cup \delta^+(Z)\\ & = & (\delta^+(X_1|X_2)Y \cup \delta^+(Y))Z \cup \delta^+(Z)\\ & \overset{(\star)}{=} & \delta^+(X)Z \cup \delta^+(Z)\end{array}$$

In $(\star)$, if $Y = \varepsilon$, the equality holds by definition of derivatives of a choice node. If $Y \neq \varepsilon$, then $X_1|X_2$ is smaller than X, and one can apply the IH on $(X_1|X_2)Y$ with $X_1|X_2$ as X and Y as an instance of the universal variable Z in the lemma.

Induction case $X = S*Y$**.**
Then $\delta(X) = \delta(S)X\,|\,\delta(Y)$ because $S*$ is nullable. The proof step uses (1), for any normalized W:

$$\delta^+(S*W) = \delta^+(S)S*W \cup \delta^+(W) \qquad (1)$$

Equation (1) is proved first as follows:

$$\begin{array}{rcl}\delta^+(S*W) & = & \delta^+(SS*W) \cup \delta^+(W)\\ & \overset{(IH)}{=} & \delta^+(S)S*W \cup \delta^+(S*W) \cup \delta^+(W)\\ & \overset{(lfp)}{=} & \delta^+(S)S*W \cup \delta^+(W)\end{array}$$

where (lfp) holds because $\delta^+(S*W) \subseteq \delta^+(S)S*W \cup \delta^+(W)$ that can be shown separately. It follows that

$$\begin{array}{rcl}\delta^+(XZ) & = & \delta^+(S*(YZ))\\ & \overset{(1)}{=} & \delta^+(S)S*YZ \cup \delta^+(YZ)\\ & \overset{IH}{=} & \delta^+(S)S*YZ \cup \delta^+(Y)Z \cup \delta^+(Z)\\ & = & (\delta^+(S)S*Y \cup \delta^+(Y))Z \cup \delta^+(Z)\\ & \overset{(1)}{=} & \delta^+(S*Y)Z \cup \delta^+(Z)\\ & = & \delta^+(X)Z \cup \delta^+(Z)\end{array}$$

The statement follows by the induction principle. Observe that (1) implies the second part of the lemma by setting $W = \varepsilon$. □

Proof of Theorem 7.3.

Proof. If R is normalized we can use a slighltly condensed definition of $\delta(R)$ that is inlined in the proof. We prove (2)

$$|\delta^+(R)| \leq \sharp(R) \qquad (2)$$

by induction over $R = R_1\cdot Z$ where R_1 is not a concatenation and possibly $Z = \varepsilon$, corresponding to the case that R is not a concatenation or that R is a conjuction or complement.

Base case $R = \varepsilon$**.** Then $|\delta^+(R)| = 0 = \sharp(R)$.

Induction case $R = \psi Z$**.** Then $\delta(\psi Z) = \mathbf{IF}(\psi, Z, \bot)$ and so $\delta^+(R) = \{Z\} \cup \delta^+(Z)$. Here $Z \in RE$ counts for one terminal and ψ counts for one predicate node. Thus

$$|\delta^+(R)| = |\delta^+(Z)| + 1 \overset{\text{IH}}{\leq} \sharp(Z) + 1 = \sharp(R).$$

Induction case $R = (X|Y)Z$**.** Then $\delta(R) = \delta(XZ)\ |\ \delta(YZ)$ and so $\delta^+(R) = \delta^+(XZ) \cup \delta^+(YZ)$. Then, by Lemma B.1,

$$\delta^+(R) = \delta^+(XZ) \cup \delta^+(YZ) = \delta^+(X)Z \cup \delta^+(Y)Z \cup \delta^+(Z)$$

which implies that (observe that, for a set S, $|S\cdot Z| = |S|$)

$$|\delta^+(R)| \leq |\delta^+(X)| + |\delta^+(Y)| + |\delta^+(Z)| \overset{\text{IH}}{\leq} \sharp(X) + \sharp(Y) + \sharp(Z) = \sharp(R).$$

Induction case $R = S*Z$**.** Then $\delta(R) = \delta(S)S*Z\ |\ \delta(Z)$ and so, by using Lemma B.1, $\delta^+(R) = \delta^+(S)S*Z \cup \delta^+(Z)$. Then

$$|\delta^+(R)| \leq |\delta^+(S)| + |\delta^+(Z)| \overset{\text{IH}}{\leq} \sharp(S) + \sharp(Z) = \sharp(R).$$

Induction case $R = (X\&Y)$**.** Then $\delta(R) = \delta(X)\&\delta(Y)$ and thus $\delta^+(R) = \delta^+(X) \cup \delta^+(Y)$. It follows that

$$|\delta^+(R)| \leq |\delta^+(X)| + |\delta^+(Y)| \overset{\text{IH}}{\leq} \sharp(X) + \sharp(Y) = \sharp(R).$$

	Solver	Time (s)						Result					
		< .04	< .12	< .37	< 1.1	< 3.3	< 10	sat	unsat	unchk	wrong	tmout	err
Kaluza	dz3	5018	71	48	22	15	10	2608	2576	0	0	268	0
	z3	4325	582	77	30	38	47	2521	2578	0	0	353	0
	z3str3	4439	569	241	22	33	6	2728	2577	5	0	127	15
	z3trau	3998	728	259	104	63	96	2657	2591	0	0	204	0
	cvc4	3744	1323	62	183	6	122	2849	2591	0	0	12	0
	ostrich	0	0	0	1747	2369	65	1665	2516	0	0	0	1271
Slog	dz3	1884	55	23	3	1	0	798	1168	0	0	10	0
	z3	934	101	4	31	30	36	71	1065	0	0	840	0
	z3str3	1143	542	89	36	25	21	784	1072	0	0	120	0
	z3trau	1224	178	186	138	106	61	727	1166	0	1	82	0
	cvc4	1887	61	24	4	0	0	808	1168	0	0	0	0
	ostrich	0	0	0	1363	583	21	800	1167	0	0	1	8
Norn	dz3	366	282	69	13	2	0	594	138	0	0	81	0
	z3	76	103	98	124	51	90	469	73	0	0	274	0
	z3str3	626	2	0	1	0	0	567	62	0	0	187	0
	z3trau	170	78	5	1	1	0	208	47	0	115	0	446
	cvc4	544	132	27	2	30	5	591	149	0	0	73	3
	ostrich	0	0	0	439	377	0	597	219	0	0	0	0
Norn	dz3	82	13	3	1	0	0	67	32	0	0	48	0
	z3	44	30	9	6	3	2	63	31	0	0	53	0
	z3str3	77	0	0	0	0	0	60	17	0	0	70	0
	z3trau	47	50	4	0	0	0	34	67	0	27	0	19
	cvc4	96	25	3	3	1	0	66	62	0	0	19	0
	ostrich	0	0	0	90	57	0	67	80	0	0	0	0
SyGuS-qgen	dz3	126	176	41	0	0	0	331	0	12	0	0	0
	z3	0	0	0	0	14	51	65	0	0	0	278	0
	z3str3	277	4	0	0	0	0	273	0	8	0	41	21
	z3trau	0	0	8	51	24	120	201	0	2	0	105	35
	cvc4	21	17	124	102	62	7	333	0	0	0	10	0
	ostrich	0	0	0	0	0	0	0	0	0	0	0	343
RegExLib Intersection	dz3	6	9	14	4	2	0	26	9	0	0	20	0
	z3	1	2	3	12	9	0	4	23	0	0	28	0
	z3str3	2	1	6	12	6	0	4	23	0	0	28	0
	z3trau	2	0	4	12	8	0	3	23	0	0	29	0
	cvc4	2	9	4	1	1	3	20	0	0	0	35	0
	ostrich	0	0	0	11	34	0	25	20	0	0	0	10
RegExLib Subset	dz3	26	28	27	5	5	0	90	1	0	0	9	0
	z3	0	0	0	2	5	3	7	3	0	0	90	0
	z3str3	0	0	0	2	4	3	6	3	0	0	91	0
	z3trau	0	0	0	0	5	4	6	3	0	0	91	0
	cvc4	17	46	12	2	1	2	80	0	0	0	20	0
	ostrich	0	0	0	12	69	0	72	9	0	0	0	19
Handwr.	dz3	35	14	7	9	7	6	42	36	0	0	10	1
	z3	20	4	4	6	2	1	14	23	0	2	46	4
	cvc4	28	6	3	2	7	5	28	23	0	0	25	13
	ostrich	0	0	0	52	24	0	40	36	0	5	6	2

Figure 6. Full results of the experiments, divided by double lines into non-Boolean benchmarks (regular expression constraints are on separate variables, top), Boolean benchmarks (multiple regular expression constraints on the same variable, middle), and additional handcrafted Boolean examples (bottom).

Induction case $R = {\sim}X$. Here $\delta(R) = {\sim}\delta(X)$. It follows that

$$|\delta^+(R)| = |\delta^+(X)| \overset{\text{IH}}{\leq} \sharp(X) = \sharp(R).$$

Equation (2) follows by the induction principle So $Q_{\text{SBFA}(R)} = \delta^+(R) \cup \{R, \bot, .*\}$, where $.* = {\sim}\bot$, and, by (2), $|Q_{\text{SBFA}(R)}| \leq |\delta^+(R)| + 3 \leq \sharp(R) + 3$. □

C More on Relation to AFAs and BFAs

Algebra $\mathcal{A}$ is assumed to be such that Σ is finite and for each $a \in \Sigma$ there is a predicate $\hat{a}$ such that $[\![\hat{a}]\!] = \{a\}$. In a pure classical setting of finite automata, transition functions are usually parameterized by single characters, so the notion of character predicates seems vacuous. In our translation below we will make use of $\mathcal{A}$, where input predicates such as $\neg\hat{a}$ arise implicitly, because for example, a transition predicate ${\sim}\mathbf{ite}(\hat{a}, q, \bot)$ simplifies to $\mathbf{ite}(\hat{a}, \overline{q}, \overline{q_\bot})$ that logically corresponds to $\mathbf{ite}(\hat{a}, {\sim}q, \bot) \,|\, \mathbf{ite}(\neg\hat{a}, q_\top, \bot)$. Perhaps the most common use of predicates is that $\mathbf{ite}(\alpha, q, \bot) \,|\, \mathbf{ite}(\beta, q, \bot)$ reduces to $\mathbf{ite}(\alpha \vee \beta, q, \bot)$, and, analogously, $\mathbf{ite}(\alpha, q, \bot) \& \mathbf{ite}(\beta, q, \bot)$ reduces to $\mathbf{ite}(\alpha \wedge \beta, q, \bot)$.

AFA. Alternating finite automata [13, 28] (AFAs) have a finite *input alphabet* Σ, a finite *set of states* $Q = \{q_i\}_{i=0}^{k-1}$, a *start state* $\iota \in Q$, a set of *final states* $F \subseteq Q$, and a *transition function* $g : Q \to (\Sigma \times \{0,1\}^{(k)}) \to \{0,1\}$. Let $g_p = g(p)$ for $p \in Q$. A sequence $v \in \{0,1\}^{(k)}$ represents the *conjuction*

$$\mathbf{q}_v = AND\{q_i \in Q \mid v_i = 1\}$$

and for $a \in \Sigma$, $p \in Q$, $\lambda v.g_p(a, v)$ represents the *disjunction*

$$g_{p,a} = OR\{\mathbf{q}_v \mid g_p(a, v) = 1, v \in \{0,1\}^{(k)}\},$$

where $OR(\emptyset) = q_\bot$ is a new state and $AND(\emptyset) = {\sim}q_\bot$. The translation of $M_{\text{AFA}} = (Q, \Sigma, \iota, F, g)$ into an equivalent SBFA is as follows

$$SBFA(M_{\text{AFA}}) = \big(\mathcal{A}, Q \cup \{q_\bot\}, \iota, F, q_\bot, \{q_\bot \mapsto q_\bot\} \bigcup_{p \in Q} \{p \mapsto OR_{a \in \Sigma} \mathbf{ite}(\hat{a}, g_{p,a}, q_\bot)\}\big)$$

Proposition C.1. $\mathbf{L}(SBFA(M_{AFA})) = L(M_{AFA})$ *with* L *as in [13].*

BFA. BFAs over Σ have a finite *set of states* Q an *initial function* $\iota \in \mathbb{B}(Q)$, a set of *final states* $F \subseteq Q$, and a *transition function* $\delta : Q \times \Sigma \to \mathbb{B}(Q)$.

We translate $M_{\text{BFA}} = (\Sigma, Q, \delta, \iota, F)$ into an equivalent SBFA as follows. The translation uses the fresh state $q_\bot \notin Q$.

$$SBFA(M_{\text{BFA}}) = (\mathcal{A}, Q \cup \{q_\bot\}, \iota, F, q_\bot, \{q_\bot \mapsto q_\bot\} \bigcup_{p \in Q} \{p \mapsto OR_{a \in \Sigma} \mathbf{ite}(\hat{a}, \delta(p, a), q_\bot)\})$$

where $\hat{a}$ is the predicate in $\mathcal{A}$ such that $[\![\hat{a}]\!] = \{a\}$.

Proposition C.2. $\mathbf{L}(SBFA(M_{BFA})) = L(M_{BFA})$ *with* L *as in [10].*

Remarks. Observe that the main difference between M_{AFA} and M_{BFA} besides the initial state formula is that g relies essentially on $DNF(\mathbb{B}^+(Q))$ while δ uses the full $\mathbb{B}(Q)$ for state predicates. In that respect, the BFA formulation is closer in spirit to SBFAs. Thus, because of DNF, the size of δ can be exponentially more succinct than g (*if* g is represented explicitly as its type suggests). Negation does not play a big role here because it can be eliminated at a linear cost. Therefore, AFAs and BFAs are to a large extent considered to be equivalent notions. As we know, this is not true in the symbolic case, when comparing SAFAs and SBFAs.

D Lift rules

The lifting rule $lift(\tau)$ propagates the intersection into the leaves and thus lifts the conditionals to the top level. Here we also pass the branch condition ψ that is initally ., that can be maintained to be satisfiable, so that dead branches are eliminated on-the-fly and the resulting transition regex is *clean* — in all conditional regexes all branches are satisfiable. Assume here that τ is in NNF. The NNF rules are specified below.

$$\begin{aligned} lift(\tau) &= lift_{.}(\tau) \\ lift_\psi(\tau) &= \bot \quad \text{if } \psi \equiv \bot \end{aligned}$$

In the remainder ψ is assumed satisfiable ($\psi \not\equiv \bot$).

$$\begin{aligned} lift_\psi(R) &= R \quad \text{if } R \in ERE \text{ and } \psi \equiv . \\ lift_\psi(R) &= \mathbf{ite}(\psi, R, \bot) \quad \text{if } R \in ERE \text{ and } \psi \not\equiv . \\ lift_\psi(\mathbf{ite}(\varphi, t, f)) &= \mathbf{ite}(\varphi, lift_{\psi \wedge \varphi}(t), lift_{\psi \wedge \neg\varphi}(f)) \\ lift_\psi(\mathbf{ite}(\varphi, t, f) \,\&\, \rho) &= lift_\psi(\mathbf{ite}(\varphi, t \,\&\, \rho, f \,\&\, \rho)) \\ lift_\psi((\tau_1 \mid \tau_2) \,\&\, \rho) &= lift_\psi(\tau_1 \,\&\, \rho) \mid lift_\psi(\tau_2 \,\&\, \rho) \end{aligned}$$

NNF. The *negation normal form* of a transition regex τ, $NNF(\tau)$, is defined as follows. The correctness of these rules rests on Lemma 4.2.

$$\begin{aligned} NNF(\mathbf{ite}(\varphi, \tau, \rho)) &= \mathbf{ite}(\varphi, NNF(\tau), NNF(\rho)) \\ NNF({\sim}\mathbf{ite}(\varphi, \tau, \rho)) &= \mathbf{ite}(\varphi, NNF({\sim}\tau), NNF({\sim}\rho)) \\ NNF({\sim}{\sim}\tau) &= NNF(\tau) \\ NNF({\sim}R) &= {\sim}R \quad \text{if } R \in ERE \end{aligned}$$

The remaining cases are standard applications of DeMorgan's rules.