About Top in Kleene Algebra

Damien Pous

Plume group, CNRS, ENS de Lyon

Aix-la-chapelle, 25.o9.2o25

based on joint work with
Paul Brunet, Amina Doumane, Denis Kuperberg,
Jurriaan Rot, Jana Wagemaker

Kleene algebra

Theorem

For all regular expressions e, f,

$$
\begin{array}{rrl}
 & \mathrm{KA} \vdash e = f & \text{(axiomatic proof)} \\
\Leftrightarrow & [e] = [f] & \text{(language equality)} \\
\Leftrightarrow & \mathrm{LANG} \models e = f & \text{(language models)} \\
\Leftrightarrow & \mathrm{REL} \models e = f & \text{(relational models)}
\end{array}
$$

[Kleene '56, Conway '71, Kozen '91, Krob '91]

Completeness is hard!

Kleene algebra

Theorem

For all regular expressions e, f,

$$
\begin{aligned}
& & \mathrm{KA} \vdash e = f & \quad \text{(axiomatic proof)} \\
&\Leftrightarrow & [e] = [f] & \quad \text{(language equality)} \\
&\Leftrightarrow & \mathrm{LANG} \models e = f & \quad \text{(language models)} \\
&\Leftrightarrow & \mathrm{REL} \models e = f & \quad \text{(relational models)}
\end{aligned}
$$

[Kleene '56, Conway '71, Kozen '91, Krob '91]

Completeness is hard!

Yield tools for reasoning about relations:

- algebraically
- automatically

A property of relations

$$(S^\star \cdot R \cdot S^\star)^+ = (R \cup S)^\star \cdot R \cdot (R \cup S)^\star$$

A property of relations

$$(S^\star \cdot R \cdot S^\star)^+ = (R \cup S)^\star \cdot R \cdot (R \cup S)^\star$$

```
text {* Adapted from C.Sternagel and R.Thiemann.
  Cf. http://afp.hg.sourceforge.net/hgweb/afp/afp/rev/dbb9a8a88678
*}

lemma relto_trancl_conv: "(S^* O R O S^*)^+ = (R ∪ S)^* O R O (R ∪ S)^*"
proof
  show "(R ∪ S)^* O R O (R ∪ S)^* ⊆ (relto R S)^+"
  proof(clarify, simp)
    fix x1 x2 x3 x4
    assume x12: "(x1,x2) ∈ (R ∪ S)^*" and x23: "(x2,x3) ∈ R" and x34: "(x3,x4) ∈ (R ∪ S)^*"
    let ?S = "S^*"
    {
      fix x y z
      assume "(y,z) ∈ (R ∪ S)^*"
      hence "(x,y) ∈ relto R S ⟶ (x,z) ∈ (relto R S)^+"
      proof (induct)
        case base
        show ?case by auto
      next
        case (step u z)
        show ?case
        proof
          assume "(x,y) ∈ relto R S"
          with step have nearly: "(x,u) ∈ (relto R S)^+" by simp
          from step(2)
          show "(x,z) ∈ (relto R S)^+"
          proof
            assume "(u,z) ∈ R"
            hence "(u,z) ∈ relto R S" by auto
            with nearly show ?thesis by auto
          next
            assume uz: "(u,z) ∈ S"
            from nearly[unfolded trancl_unfold_right]
            obtain v where xv: "(x,v) ∈ (relto R S)^*" and vu: "(v,u) ∈ relto R S" by auto
            from vu obtain w where vw: "(v,w) ∈ S^* O R" and wu: "(w,u) ∈ S^*" by auto
            from wu uz have wz: "(w,z) ∈ S^*" by auto
            with vw have vz: "(v,z) ∈ relto R S" by auto
            with xv show ?thesis by auto
          qed
        qed
      qed
    } note steps_right = this
    from x23 have "(x2,x3) ∈ relto R S" by auto
    from mp[OF steps_right[OF x34] this] have x24: "(x2,x4) ∈ (relto R S)^+" .
    with x12 show "(x1,x4) ∈ (relto R S)^+"
    proof (induct arbitrary: x4, simp)
      case (step y z)
      from step(2)
      have "(y,x4) ∈ (relto R S)^+"
      proof
        assume "(y,z) ∈ R"
        hence "(y,z) ∈ relto R S" by auto
        with step(4) show ?thesis by auto
      next
        assume yz: "(y,z) ∈ S"
        from step(4)[unfolded trancl_unfold_left]
        obtain v where zv: "(z,v) ∈ relto R S" and vx4: "(v,x4) ∈ (relto R S)^*" by auto
        from zv obtain w where zw: "(z,w) ∈ S^*" and wv: "(w,v) ∈ R O S^*" by auto
        from yz zw have "(y,w) ∈ S^*" by auto
        with wv have "(y,v) ∈ relto R S" by auto
        with vx4 show ?thesis by auto
      qed
      from step(3)[OF this] show ?case .
    qed
  qed
next
  have S: "S^* ⊆ (R ∪ S)^*" by (rule rtrancl_mono[of S "R ∪ S", simplified])
  have R: "R ⊆ (R ∪ S)^*" by auto
  show "(relto R S)^+ ⊆ (R ∪ S)^* O R O (R ∪ S)^*"
  proof (rule subrelI)
    fix x y
    assume "(x,y) ∈ (S^* O R O S^*)^+"
    thus "(x,y) ∈ (R ∪ S)^* O R O (R ∪ S)^*"
    proof (induct)
      case (base y)
      thus ?case using S by blast
    next
      case (step y z)
      from step(2) have "(y,z) ∈ (R ∪ S)^* O (R ∪ S)^* O (R ∪ S)^*" using R S by blast
      hence "(y,z) ∈ (R ∪ S)^*" by auto
      with step (3) show ?case by force
    qed
  qed
qed
```

A property of relations

$$(S^\star \cdot R \cdot S^\star)^+ = (R \cup S)^\star \cdot R \cdot (R \cup S)^\star$$

```
text {* Adapted from C.Sternagel and R.Thiemann.
  Cf. http://afp.hg.sourceforge.net/hgweb/afp/afp/rev/dbb9a8a88678
*}

lemma relto_trancl_conv: "(S^* O R O S^*)^+ = (R ∪ S)^* O R O (R ∪ S)^*"
proof
  show "(R ∪ S)^* O R O (R ∪ S)^* ⊆ (relto R S)^+"
  proof(clarify, simp)
    fix x1 x2 x3 x4
    assume x12: "(x1,x2) ∈ (R
    let ?S = "S^*"
    {
      fix x y z
      assume "(y,z) ∈ (R ∪ S)^
      hence "(x,y) ∈ relto R S
      proof (induct)
        case base
        show ?case by auto
      next
        case (step u z)
        show ?case
        proof
          assume "(x,y) ∈ relto R S"
          with step have nearly: "(x,u) ∈ (relto R S)^+" by simp
          from step(2)
          show "(x,z) ∈ (relto R S)^+"
          proof
            assume "(u,z) ∈ R"
            hence "(u,z) ∈ relto R S" by auto
            with nearly show ?thesis by auto
          next
            assume uz: "(u,z) ∈ S"
            from nearly[unfolded trancl_unfold_right]
            obtain v where xv: "(x,v) ∈ (relto R S)^*" and vu: "(v,u) ∈ relto R S" by auto
            from vu obtain w where vw: "(v,w) ∈ S^* O R" and wu: "(w,u) ∈ S^*" by auto
            from wu uz have wz: "(w,z) ∈ S^*" by auto
            with vw have vz: "(v,z) ∈ relto R S" by auto
            with xv show ?thesis by auto
          qed
        qed
      qed
    } note steps_right = this
```

```
    from x23 have "(x2,x3) ∈ relto R S" by auto
    from mp[OF steps_right[OF x34] this] have x24: "(x2,x4) ∈ (relto R S)^+" .
    with x12 show "(x1,x4) ∈ (relto R S)^+"
    proof (induct arbitrary: x4, simp)
      case (step y z)
      from step(2)
      have "(y,x4) ∈ (relto R S)^+"
      proof
eft]
 and vx4: "(v,x4) ∈ (relto R S)^*" by auto
^*" and wv: "(w,v) ∈ R O S^*" by auto
uto
      qed
      from step(3)[OF this] show ?case .
    qed
  qed
next
  have S: "S^* ⊆ (R ∪ S)^*" by (rule rtrancl_mono[of S "R ∪ S", simplified])
  have R: "R ⊆ (R ∪ S)^*" by auto
  show "(relto R S)^+ ⊆ (R ∪ S)^* O R O (R ∪ S)^*"
  proof (rule subrelI)
    fix x y
    assume "(x,y) ∈ (S^* O R O S^*)^+"
    thus "(x,y) ∈ (R ∪ S)^* O R O (R ∪ S)^*"
    proof (induct)
      case (base y)
      thus ?case using S by blast
    next
      case (step y z)
      from step(2) have "(y,z) ∈ (R ∪ S)^* O (R ∪ S)^* O (R ∪ S)^*" using R S by blast
      hence "(y,z) ∈ (R ∪ S)^*" by auto
      with step (3) show ?case by force
    qed
  qed
qed
```

```
Lemma relto_trancl_conv: forall R S,
   (S^* o R o S^*)^+ == (R + S)^* o R o (R + S)^*
Proof. ka. Qed.
```

Paterson's flowchart equivalence [Manna'74]

of y_2 is not used in statement 5, we can execute statement 4 after testing $p(y_1)$; similarly, since the value of y_1 is not used in statement 8, we can execute statement 6 after testing $p(y_2)$.

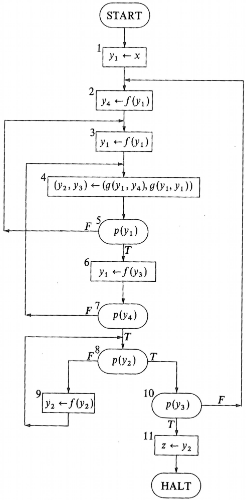


Figure 4-6a Schema S_{6A}.

. . .

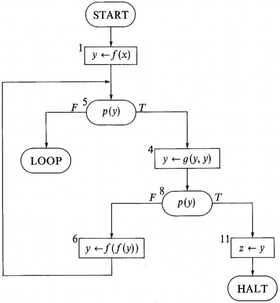


Figure 4-6e Schema S_{6E}

Step 4: $S_{6D} \approx S_{6E}$. Considering statements 4 and 6 in S_{6D}, we realize that y_2 is merely a dummy variable and can be replaced by y_1. Therefore, dropping the subscript and modifying statements 1, 3, and 6, we obtain S_{6E}. □

Paterson's flowchart equivalence [Manna'74]

of y_2 is not used in statement 5, we can execute statement 4 after testing $p(y_1)$; similarly, since the value of y_1 is not used in statement 8, we can execute statement 6 after testing $p(y_2)$.

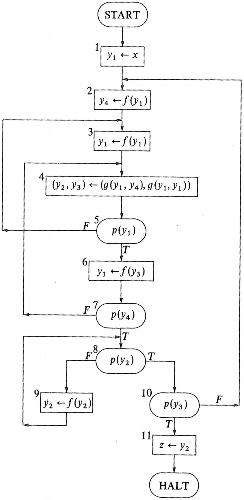


Figure 4-6a Schema S_{6A}.

. . .

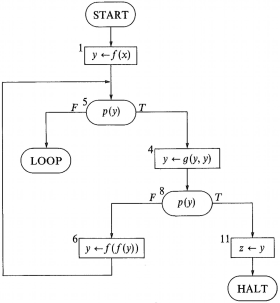


Figure 4-6e Schema S_{6E}

Step 4: $S_{6D} \approx S_{6E}$. Considering statements 4 and 6 in S_{6D}, we realize that y_2 is merely a dummy variable and can be replaced by y_1. Therefore, dropping the subscript and modifying statements 1, 3, and 6, we obtain S_{6E}. □

one page of plain english text
three pages for intermediate flowcharts

Paterson's flowchart equivalence [Angus and Kozen'01]

Kleene Algebra with Tests and Program Schematology

Allegra Angus Dexter Kozen

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501, USA

July 10, 2001

Abstract

The theory of flowchart schemes has a rich history going back to Ianov [6]; see Manna [22] for an elementary exposition. A central question in the theory of program schemes is scheme equivalence. Manna presents several examples of equivalence proofs that work by simplifying the schemes using various combinatorial transformation rules. In this paper we present a purely algebraic approach to this problem using Kleene algebra with tests (KAT). Instead of transforming schemes directly using combinatorial graph manipulation, we regard them as a certain kind of automaton on abstract traces. We prove a generalization of Kleene's theorem and use it to construct equivalent expressions in the language of KAT. We can then give a purely equational proof of the equivalence of the resulting expressions. We prove soundness of the method and give a detailed example of its use.

six pages of KAT computations

Paterson's flowchart equivalence [In Rocq]

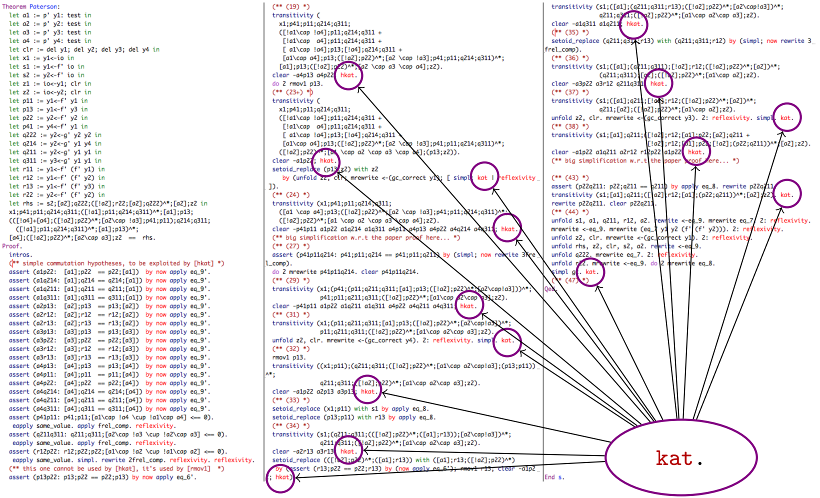
Paterson's flowchart equivalence [In Rocq]

```
Theorem Paterson:
  let a1 := p' y1: test in
  let a2 := p' y2: test in
  let a3 := p' y3: test in
  let a4 := p' y4: test in
  let clr := del y1; del y2; del y3; del y4 in
  let x1 := y1<-io in
  let s1 := y1<-f' io in
  let s2 := y2<-f' io in
  let z1 := io<-y1; clr in
  let z2 := io<-y2; clr in
  let p11 := y1<-f' y1 in
  let p13 := y1<-f' y3 in
  let p22 := y2<-f' y2 in
  let p41 := y4<-f' y1 in
  let q222 := y2<-g' y2 y2 in
  let q214 := y2<-g' y1 y4 in
  let q211 := y2<-g' y1 y1 in
  let q311 := y3<-g' y1 y1 in
  let r11 := y1<-f' (f' y1) in
  let r12 := y1<-f' (f' y2) in
  let r13 := y1<-f' (f' y3) in
  let r22 := y2<-f' (f' y2) in
  let rhs := s2;[a2];q222;([!a2];r22;[a2];q222)^*;[a2];z2 in
  x1;p41;p11;q214;q311;([!a1];p11;q214;q311)^*;[a1];p13;
  (([!a4]+[a4];([!a2];p22)^*;[a2\cap !a3];p41;p11);q214;q311;
    ([!a1];p11;q214;q311)^*;[a1];p13)^*;
  [a4];([!a2];p22)^*;[a2\cap a3];z2  ==  rhs.
Proof.
  intros.
  (** simple commutation hypotheses, to be exploited by [hkat] *)
  assert (a1p22:  [a1];p22  == p22;[a1])  by now apply eq_9'.
  assert (a1q214: [a1];q214 == q214;[a1]) by now apply eq_9'.
  assert (a1q211: [a1];q211 == q211;[a1]) by now apply eq_9'.
  assert (a1q311: [a1];q311 == q311;[a1]) by now apply eq_9'.
  assert (a2p13:  [a2];p13  == p13;[a2])  by now apply eq_9'.
  assert (a2r12:  [a2];r12  == r12;[a2])  by now apply eq_9'.
  assert (a2r13:  [a2];r13  == r13;[a2])  by now apply eq_9'.
  assert (a3p13:  [a3];p13  == p13;[a3])  by now apply eq_9'.
  assert (a3p22:  [a3];p22  == p22;[a3])  by now apply eq_9'.
  assert (a3r12:  [a3];r12  == r12;[a3])  by now apply eq_9'.
  assert (a3r13:  [a3];r13  == r13;[a3])  by now apply eq_9'.
  assert (a4p13:  [a4];p13  == p13;[a4])  by now apply eq_9'.
  assert (a4p11:  [a4];p11  == p11;[a4])  by now apply eq_9'.
  assert (a4p22:  [a4];p22  == p22;[a4])  by now apply eq_9'.
  assert (a4q214: [a4];q214 == q214;[a4]) by now apply eq_9'.
  assert (a4q211: [a4];q211 == q211;[a4]) by now apply eq_9'.
  assert (a4q311: [a4];q311 == q311;[a4]) by now apply eq_9'.
  assert (p41p11: p41;p11;[a1\cap !a4 \cup !a1\cap a4] <== 0).
   eapply same_value. apply frel_comp. reflexivity.
  assert (q211q311: q211;q311;[a2\cap !a3 \cup !a2\cap a3] <== 0).
   eapply same_value. apply frel_comp. reflexivity.
  assert (r12p22: r12;p22;p22;[a1\cap !a2 \cup !a1\cap a2] <== 0).
   eapply same_value. simpl. rewrite 2frel_comp. reflexivity. reflexivity.
  (** this one cannot be used by [hkat], it's used by [rmov1]  *)
  assert (p13p22: p13;p22 == p22;p13) by now apply eq_6'.
  (** (19) *)
  transitivity (
    x1;p41;p11;q214;q311;
    ([!a1\cap !a4];p11;q214;q311 +
     [!a1\cap  a4];p11;q214;q311 +
     [ a1\cap !a4];p13;[!a4];q214;q311 +
     [a1\cap a4];p13;([!a2];p22)^*;[a2 \cap !a3];p41;p11;q214;q311)^*;
    [a1];p13;([!a2];p22)^*;[a2 \cap a3 \cap a4];z2).
  clear -a4p13 a4p22. hkat.
  do 2 rmov1 p13.
  (** (23+) *)
  transitivity (
    x1;p41;p11;q214;q311;
    ([!a1\cap !a4];p11;q214;q311 +
     [!a1\cap  a4];p11;q214;q311 +
     [ a1\cap !a4];p13;[!a4];q214;q311 +
     [a1\cap a4];p13;([!a2];p22)^*;[a2 \cap !a3];p41;p11;q214;q311)^*;
    ([!a2];p22)^*;[a1 \cap a2 \cap a3 \cap a4];(p13;z2)).
  clear -a1p22; hkat.
  setoid_replace (p13;z2) with z2
     by (unfold z2, clr; mrewrite <-(gc_correct y1); [ simpl; kat | reflexivity ]).
  (** (24) *)
  transitivity (x1;p41;p11;q214;q311;
    ([a1 \cap a4];p13;([!a2];p22)^*;[a2 \cap !a3];p41;p11;q214;q311)^*;
    ([!a2];p22)^*;[a1 \cap a2 \cap a3 \cap a4];z2).
  clear -p41p11 a1p22 a1q214 a1q311 a4p11 a4p13 a4p22 a4q214 a4q311; hkat.
  (** big simplification w.r.t the paper proof here... *)
  (** (27) *)
  assert (p41p11q214: p41;p11;q214 == p41;p11;q211) by (simpl; now rewrite 3frel_comp).
  do 2 mrewrite p41p11q214. clear p41p11q214.
  (** (29) *)
  transitivity (x1;(p41;(p11;q211;q311;[a1];p13;([!a2];p22)^*;[a2\cap!a3]))^*;
                p41;p11;q211;q311;([!a2];p22)^*;[a1\cap a2\cap a3];z2).
  clear -p41p11 a1p22 a1q211 a1q311 a4p22 a4q211 a4q311; hkat.
  (** (31) *)
  transitivity (x1;(p11;q211;q311;[a1];p13;([!a2];p22)^*;[a2\cap!a3])^*;
                p11;q211;q311;([!a2];p22)^*;[a1\cap a2\cap a3];z2).
  unfold z2, clr. mrewrite <-(gc_correct y4). 2: reflexivity. simpl. kat.
  (** (32) *)
  rmov1 p13.
  transitivity ((x1;p11);(q211;q311;([!a2];p22)^*;[a1\cap a2\cap!a3];(p13;p11))^*;
                q211;q311;([!a2];p22)^*;[a1\cap a2\cap a3];z2).
  clear -a1p22 a2p13 a3p13; hkat.
  (** (33) *)
  setoid_replace (x1;p11) with s1 by apply eq_8.
  setoid_replace (p13;p11) with r13 by apply eq_8.
  (** (34) *)
  transitivity (s1;(q211;q311;(([!a2];p22)^*;([a1];r13));[a2\cap!a3])^*;
                q211;q311;([!a2];p22)^*;[a1\cap a2\cap a3];z2).
  clear -a2r13 a3r13; hkat.
  setoid_replace (([!a2];p22)^*;([a1];r13)) with ([a1];r13;([!a2];p22)^*)
   by (assert (r13;p22 == p22;r13) by (now apply eq_6'); rmov1 r13; clear -a1p22; hkat).
  transitivity (s1;([a1];(q211;q311;r13);([!a2];p22)^*;[a2\cap!a3])^*;
                q211;q311;([!a2];p22)^*;[a1\cap a2\cap a3];z2).
  clear -a1q311 a1q211; hkat.
  (** (35) *)
  setoid_replace (q211;q311;r13) with (q211;q311;r12) by (simpl; now rewrite 3frel_comp).
  (** (36) *)
  transitivity (s1;([a1];(q211;q311);[!a2];r12;([!a2];p22)^*;[a2])^*;
                (q211;q311);[a2];([!a2];p22)^*;[a1\cap a2];z2).
  clear -a3p22 a3r12 q211q311. hkat.
  (** (37) *)
  transitivity (s1;([a1];q211;[!a2];r12;([!a2];p22)^*;[a2])^*;
                q211;[a2];([!a2];p22)^*;[a1\cap a2];z2).
  unfold z2, clr. mrewrite <-(gc_correct y3). 2: reflexivity. simpl. kat.
  (** (38) *)
  transitivity (s1;[a1];q211;([!a2];r12;[a1];p22;[a2];q211 +
                             [!a2];r12;[a1];p22;[!a2];(p22;q211))^*;[a2];z2).
  clear -a1p22 a1q211 a2r12 r12p22 a1p22. hkat.
  (** big simplification w.r.t the paper proof here... *)

  (** (43) *)
  assert (p22q211: p22;q211 == q211) by apply eq_8. rewrite p22q211.
  transitivity (s1;[a1];q211;([!a2];r12;[a1];(p22;q211))^*;[a2];z2). kat.
  rewrite p22q211. clear p22q211.
  (** (44) *)
  unfold s1, a1, q211, r12, a2. rewrite <-eq_9. mrewrite eq_7. 2: reflexivity.
  mrewrite <-eq_9. mrewrite (eq_7 y1 y2 (f' (f' y2))). 2: reflexivity.
  unfold z2, clr. mrewrite <-(gc_correct y1). 2: reflexivity.
  unfold rhs, z2, clr, s2, a2. rewrite <-eq_9.
  unfold q222. mrewrite eq_7. 2: reflexivity.
  unfold r22. mrewrite <-eq_9. do 2 mrewrite eq_8.
  simpl gc. kat.
  (** (47) *)
Qed.

End s.
```

Paterson's flowchart equivalence [In Rocq]



Paterson's flowchart equivalence [In Rocq]

The assignments to y_3 are now useless and can be eliminated by Lemma 4.5, giving

$$s_1(a_1q_{211}\bar{a}_2r_{12}(\bar{a}_2p_{22})^*a_2)^*q_{211}a_2(\bar{a}_2p_{22})^*a_1a_2z_2 \tag{38}$$

Furthermore, because of the preguard $\bar{a}_2$ and postguard a_2, the loop $(\bar{a}_2p_{22})^*a_2$ occurring inside the outer loop of (38) must be executed at least once. Similarly, because of the preguard a_2, the loop $(\bar{a}_2p_{22})^*a_2$ occurring outside the outer loop cannot be executed at all. Formally,

$$\begin{aligned}
\bar{a}_2(\bar{a}_2p_{22})^*a_2 &= \bar{a}_2a_2 + \bar{a}_2\bar{a}_2p_{22}(\bar{a}_2p_{22})^*a_2 \\
&= \bar{a}_2p_{22}(\bar{a}_2p_{22})^*a_2, \\
a_2(\bar{a}_2p_{22})^*a_2 &= a_2a_2 + a_2\bar{a}_2p_{22}(\bar{a}_2p_{22})^*a_2 \\
&= a_2.
\end{aligned}$$

Thus we can rewrite (38) as

$$s_1(a_1q_{211}r_{12}\bar{a}_2p_{22}(\bar{a}_2p_{22})^*a_2)^*q_{211}a_1a_2z_2. \tag{39}$$

Moreover, the remaining inner loop $(\bar{a}_2p_{22})^*a_2$ can be executed at most twice. To show this, we use sliding and commutativity to get

$$\begin{aligned}
&s_1a_1q_{211}(r_{12}\bar{a}_2p_{22}(\bar{a}_2p_{22})^*a_2a_1q_{211})^*a_2z_2 \\
&= s_1a_1q_{211}(\bar{a}_2r_{12}a_1p_{22}(\bar{a}_2p_{22})^*a_2q_{211})^*a_2z_2
\end{aligned} \tag{40}$$

As above,

$$\begin{aligned}
r_{12}a_1p_{22}\bar{a}_2p_{22}\bar{a}_2 &\le r_{12}a_1p_{22}p_{22}\bar{a}_2 \\
&= r_{12}a_1r_{22}\bar{a}_2 \qquad \text{by (8)} \\
&= r_{12}r_{22}a_1\bar{a}_2 \\
&= r_{12}r_{22}(a_1 \leftrightarrow a_2)a_1\bar{a}_2 \\
&= 0,
\end{aligned} \tag{41}$$

therefore

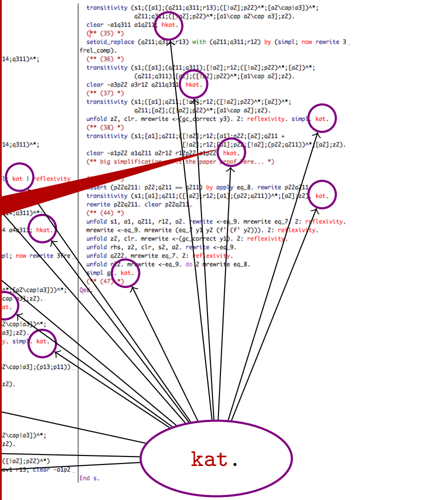
$$\begin{aligned}
&r_{12}a_1p_{22}(\bar{a}_2p_{22})^* \\
&= r_{12}a_1p_{22}(1 + \bar{a}_2p_{22} + \bar{a}_2p_{22}\bar{a}_2p_{22}(\bar{a}_2p_{22})^*) \\
&= r_{12}a_1p_{22} + r_{12}a_1p_{22}\bar{a}_2p_{22} + r_{12}a_1p_{22}\bar{a}_2p_{22}\bar{a}_2p_{22}(\bar{a}_2p_{22})^* \\
&= r_{12}a_1p_{22} + r_{12}a_1p_{22}\bar{a}_2p_{22}
\end{aligned}$$

Thus (40) is equivalent to

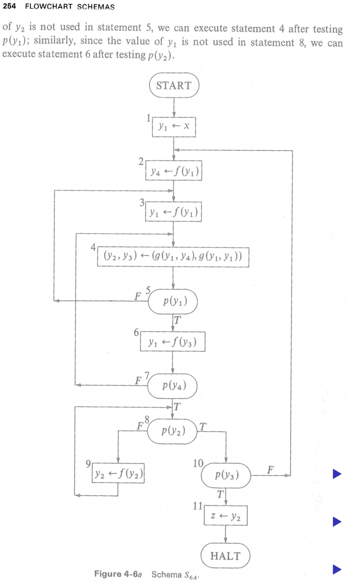
$$s_1a_1q_{211}(\bar{a}_2r_{12}a_1p_{22}a_2q_{211} + \bar{a}_2r_{12}a_1p_{22}\bar{a}_2p_{22}a_2q_{211})^*a_2z_2 \tag{42}$$

Now (41) also implies that $r_{12}a_1p_{22}\bar{a}_2p_{22} = r_{12}a_1p_{22}\bar{a}_2p_{22}a_2$, therefore (42) can be rewritten

$$s_1a_1q_{211}(\bar{a}_2r_{12}a_1p_{22}a_2q_{211} + \bar{a}_2r_{12}a_1p_{22}\bar{a}_2p_{22}q_{211})^*a_2z_2 \tag{43}$$



Paterson's flowchart equivalence [summary]

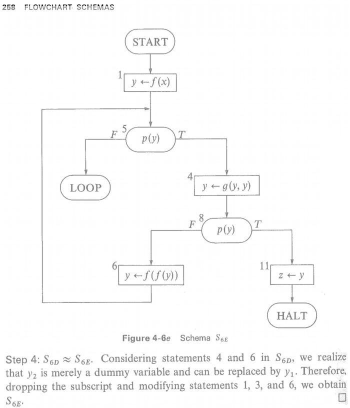


254 FLOWCHART SCHEMAS

of y_2 is not used in statement 5, we can execute statement 4 after testing $p(y_1)$; similarly, since the value of y_1 is not used in statement 8, we can execute statement 6 after testing $p(y_2)$.

Figure 4-6a Schema S_{6A}.

. . .



258 FLOWCHART SCHEMAS

Figure 4-6e Schema S_{6E}

Step 4: $S_{6D} \approx S_{6E}$. Considering statements 4 and 6 in S_{6D}, we realize that y_2 is merely a dummy variable and can be replaced by y_1. Therefore, dropping the subscript and modifying statements 1, 3, and 6, we obtain S_{6E}. □

- 1 intuitive hand-waving page in [Manna'74]
- 6 tedious pages in [Angus and Kozen'01]
- 3 formal screens in Rocq + certified KAT

What about adding a top element?

What about adding a top element?

New constant $\top$ interpreted as full language / full relation

In the context of Kleene algebra with tests (KAT), adding top makes it possible to express *incorrectness triples* [Zhang et al. '22]

What about adding a top element?

New constant $\top$ interpreted as full language / full relation

In the context of Kleene algebra with tests (KAT), adding top makes it possible to express *incorrectness triples* [Zhang et al. '22]

Some laws include:

$$\mathsf{LANG}, \mathsf{REL} \models x \leqslant \top$$

$$\mathsf{LANG}, \mathsf{REL} \models \top\top = \top$$

$$\cancel{\mathsf{LANG}}, \mathsf{REL} \models x \leqslant x \top x$$

$$\cancel{\mathsf{LANG}}, \mathsf{REL} \models \top x \top y \top = \top y \top x \top$$

What about adding a top element?

New constant $\top$ interpreted as full language / full relation

In the context of Kleene algebra with tests (KAT), adding top makes it possible to express *incorrectness triples* [Zhang et al. '22]

Some laws include:

$$\mathsf{LANG}, \mathsf{REL} \models x \leqslant \top$$

$$\mathsf{LANG}, \mathsf{REL} \models \top\top = \top$$

$$\cancel{\mathsf{LANG}}, \mathsf{REL} \models x \leqslant x\top x$$

$$\cancel{\mathsf{LANG}}, \mathsf{REL} \models \top x \top y \top = \top y \top x \top$$

The equational theories of LANG and REL differ!

We are going to prove

Theorem

For all regular expressions with top e, f*,*

	$\mathrm{KA},\ x \leqslant \top \vdash e = f$		$\mathrm{KA},\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f$
$\Leftrightarrow$	$C[e] = C[f]$	$\Leftrightarrow$	$D[e] = D[f]$
$\Leftrightarrow$	$\mathrm{LANG} \models e = f$	$\Leftrightarrow$	$\mathrm{REL} \models e = f$

We are going to prove

Theorem

For all regular expressions with top e, f,

$\mathrm{KA},\ x \leqslant \top \vdash e = f$	$\mathrm{KA},\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f$
$\Leftrightarrow \quad C[e] = C[f]$	$\Leftrightarrow \quad D[e] = D[f]$
$\Leftrightarrow \quad \mathrm{LANG} \models e = f$	$\Leftrightarrow \quad \mathrm{REL} \models e = f$

Difficulties:

- second axiom not obvious to use

$$\begin{aligned} \top x \top y \top &\leqslant \\ &\leqslant \\ &\leqslant \\ &\leqslant \top y \top x \top \end{aligned}$$

We are going to prove

Theorem

For all regular expressions with top e, f,

$$\begin{array}{rc|rc} & KA,\ x \leqslant \top \vdash e = f & & KA,\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f \\ \Leftrightarrow & C[e] = C[f] & \Leftrightarrow & D[e] = D[f] \\ \Leftrightarrow & LANG \models e = f & \Leftrightarrow & REL \models e = f \end{array}$$

Difficulties:

- second axiom not obvious to use

$$\begin{aligned} \top x \top y \top &\leqslant \top x \top y \top\ \top\ \top x \top y \top && \text{(by } z \leqslant z \top z\text{)} \\ &\leqslant \\ &\leqslant \\ &\leqslant \top y \top x \top \end{aligned}$$

We are going to prove

Theorem

For all regular expressions with top e, f*,*

	$\mathrm{KA},\ x \leqslant \top \vdash e = f$		$\mathrm{KA},\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f$
$\Leftrightarrow$	$C[e] = C[f]$	$\Leftrightarrow$	$D[e] = D[f]$
$\Leftrightarrow$	$\mathrm{LANG} \models e = f$	$\Leftrightarrow$	$\mathrm{REL} \models e = f$

Difficulties:

- second axiom not obvious to use

$$
\begin{aligned}
\top x \top y \top &\leqslant \top x \top y \top \ \top \ \top x \top y \top && \text{(by } z \leqslant z\top z\text{)}\\
&\leqslant \top x \top y \top \ \top \ \top x \top && \text{(by } z \leqslant \top\text{)}\\
&\leqslant \\
&\leqslant \top y \top x \top
\end{aligned}
$$

We are going to prove

Theorem

For all regular expressions with top e, f,

$\text{KA},\ x \leqslant \top \vdash e = f$	$\text{KA},\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f$
$\Leftrightarrow\quad C[e] = C[f]$	$\Leftrightarrow\quad D[e] = D[f]$
$\Leftrightarrow\quad \text{LANG} \models e = f$	$\Leftrightarrow\quad \text{REL} \models e = f$

Difficulties:

- second axiom not obvious to use

$$
\begin{aligned}
\top x \top y \top &\leqslant \top x \top y \top\ \top\ \top x \top y \top && (\text{by } z \leqslant z \top z)\\
&\leqslant \top x \top y \top\ \top\ \top x \top && (\text{by } z \leqslant \top)\\
&\leqslant \top x \top y \top x \top && (\text{by } z \leqslant \top)\\
&\leqslant \top y \top x \top
\end{aligned}
$$

We are going to prove

Theorem

For all regular expressions with top e, f,

$\mathrm{KA},\ x \leqslant \top \vdash e = f$	$\mathrm{KA},\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f$
$\Leftrightarrow\ C[e] = C[f]$	$\Leftrightarrow\ D[e] = D[f]$
$\Leftrightarrow\ \mathrm{LANG} \models e = f$	$\Leftrightarrow\ \mathrm{REL} \models e = f$

Difficulties:

- second axiom not obvious to use

$$
\begin{aligned}
\top x \top y \top &\leqslant \top x \top y \top\ \top\ \top x \top y \top && \text{(by } z \leqslant z\top z\text{)}\\
&\leqslant \top x \top y \top\ \top\ \top x \top && \text{(by } z \leqslant \top\text{)}\\
&\leqslant \top x \top y \top x \top && \text{(by } z \leqslant \top\text{)}\\
&\leqslant \top y \top x \top && \text{(by } z \leqslant \top\text{)}
\end{aligned}
$$

We are going to prove

Theorem

For all regular expressions with top e, f,

	$\mathsf{KA},\ x \leqslant \top \vdash e = f$		$\mathsf{KA},\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f$
$\Leftrightarrow$	$C[e] = C[f]$	$\Leftrightarrow$	$D[e] = D[f]$
$\Leftrightarrow$	$\mathsf{LANG} \models e = f$	$\Leftrightarrow$	$\mathsf{REL} \models e = f$

Difficulties:

- second axiom not obvious to use

$$\begin{aligned} \top x \top y \top &\leqslant \top x \top y \top\ \top\ \top x \top y \top && \text{(by } z \leqslant z\top z\text{)} \\ &\leqslant \top x \top y \top\ \top\ \top x \top && \text{(by } z \leqslant \top\text{)} \\ &\leqslant \top x \top y \top x \top && \text{(by } z \leqslant \top\text{)} \\ &\leqslant \top y \top x \top && \text{(by } z \leqslant \top\text{)} \end{aligned}$$

- interplay with KA reasoning

exercise: $(aaa)^\star \leqslant (aaa)^\star \top (aa)^\star + (aa)^\star a \top (aaa)^\star$

First tool: hypotheses and closed languages

Let H be a set of *hypotheses* of the shape $u \leqslant v$, with u, v words.

First tool: hypotheses and closed languages

Let H be a set of *hypotheses* of the shape $u \leqslant v$, with u, v words.
Extend H into a rewriting relation on words:

$$\leftsquigarrow_H = \{(lur, lvr) \mid l, r \in \Sigma^\star,\ (u \leqslant v) \in H\}$$

First tool: hypotheses and closed languages

Let H be a set of *hypotheses* of the shape $u \leqslant v$, with u, v words.
Extend H into a rewriting relation on words:

$$\leftsquigarrow_H = \{(lur, lvr) \mid l, r \in \Sigma^\star,\ (u \leqslant v) \in H\}$$

For a language L, define its *downward closure w.r.t. H* as:

$$C_H(L) = \{u \mid u \leftsquigarrow_H^\star v \in L\}$$

First tool: hypotheses and closed languages

Let H be a set of *hypotheses* of the shape $u \leqslant v$, with u, v words. Extend H into a rewriting relation on words:

$$\leftsquigarrow_H = \{(lur, lvr) \mid l, r \in \Sigma^\star,\ (u \leqslant v) \in H\}$$

For a language L, define its *downward closure w.r.t. H* as:

$$C_H(L) = \{u \mid u \leftsquigarrow_H^\star v \in L\}$$

Theorem (Soundness)

For all regular expressions e, f, we have

$$\mathrm{KA}, H \vdash e = f \quad \Rightarrow \quad C_H[e] = C_H[f]$$

[Doumane&Pous&Pradic&Kuperberg '19]

Converse does not always hold!

Second tool: reductions

Theorem (Completeness)

If there is a function r such that for all regular expresssions e,

1. $C_H[e] = [r(e)]$, *and*
2. $\mathsf{KA}, H \vdash e = r(e)$

Second tool: reductions

Theorem (Completeness)

If there is a function r such that for all regular expresssions e,

1. $C_H[e] = [r(e)]$, *and*
2. $\mathsf{KA}, H \vdash e = r(e)$

Then for all regular expressions e, f, we have

$$\begin{tikzcd}
\mathsf{KA}, H \vdash e = f \arrow[rr, Rightarrow, "\textit{soundness}"] & & C_H[e] = C_H[f] \arrow[d, Rightarrow, "1."] \\
\mathsf{KA} \vdash r(e) = r(f) \arrow[u, Rightarrow, "2."] & & {[r(e)] = [r(f)]} \arrow[ll, Rightarrow, "\textit{KA completeness}"]
\end{tikzcd}$$

[folklore, Pous&Rot&Wagemaker '21]

Second tool: reductions

Theorem (Completeness)

If there is a function r such that for all regular expresssions e,

1. $C_H[e] = [r(e)]$, *and*
2. $\mathsf{KA}, H \vdash e = r(e)$

Then for all regular expressions e, f, we have

$$\begin{tikzcd}
\mathsf{KA}, H \vdash e = f \arrow[r, Rightarrow, "\textit{soundness}"] & C_H[e] = C_H[f] \arrow[d, Rightarrow, "1."] \\
\mathsf{KA} \vdash r(e) = r(f) \arrow[u, Rightarrow, "2."] & {[r(e)] = [r(f)]} \arrow[l, Rightarrow, "\textit{KA completeness}"]
\end{tikzcd}$$

[folklore, Pous&Rot&Wagemaker '21]

- Example: $H = \{aa \leqslant a\}$,

Second tool: reductions

Theorem (Completeness)

If there is a function r such that for all regular expresssions e,

1. $C_H[e] = [r(e)]$, *and*
2. $\mathsf{KA}, H \vdash e = r(e)$

Then for all regular expressions e, f, we have

$$
\begin{tikzcd}
\mathsf{KA}, H \vdash e = f \arrow[r, Rightarrow, "soundness"] & C_H[e] = C_H[f] \arrow[d, Rightarrow, "1."] \\
\mathsf{KA} \vdash r(e) = r(f) \arrow[u, Rightarrow, "2."] & {[r(e)] = [r(f)]} \arrow[l, Rightarrow, "KA\ completeness"']
\end{tikzcd}
$$

[folklore, Pous&Rot&Wagemaker '21]

- Example: $H = \{aa \leqslant a\}$, set $r(a) = a^+$

Second tool: reductions

Theorem (Completeness)

If there is a function r such that for all regular expresssions e,

1. $C_H[e] = [r(e)]$, *and*
2. $\mathsf{KA}, H \vdash e = r(e)$

Then for all regular expressions e, f, we have

$$\begin{tikzcd}
\mathsf{KA}, H \vdash e = f \arrow[rr, Rightarrow, "soundness"] & & C_H[e] = C_H[f] \arrow[d, Rightarrow, "1."] \\
\mathsf{KA} \vdash r(e) = r(f) \arrow[u, Rightarrow, "2."] & & {[r(e)] = [r(f)]} \arrow[ll, Rightarrow, "KA\ completeness"]
\end{tikzcd}$$

[folklore, Pous&Rot&Wagemaker '21]

- Example: $H = \{aa \leqslant a\}$, set $r(a) = a^+$
- Non-example: $H = \{ab = ba\}$,

Second tool: reductions

Theorem (Completeness)

If there is a function r such that for all regular expresssions e,

1. $C_H[e] = [r(e)]$, *and*
2. $\mathrm{KA}, H \vdash e = r(e)$

Then for all regular expressions e, f, we have

$$\begin{tikzcd}
\mathrm{KA}, H \vdash e = f \arrow[rr, Rightarrow, "\textit{soundness}"] & & C_H[e] = C_H[f] \arrow[d, Rightarrow, "1."] \\
\mathrm{KA} \vdash r(e) = r(f) \arrow[u, Rightarrow, "2."] & & {[r(e)] = [r(f)]} \arrow[ll, Rightarrow, "\textit{KA completeness}"]
\end{tikzcd}$$

[folklore, Pous&Rot&Wagemaker '21]

- Example: $H = \{aa \leqslant a\}$, set $r(a) = a^+$
- Non-example: $H = \{ab = ba\}$, $C_H[(ab)^\star] = ??$

Languages (easy!)

Consider $\top$ as a new letter in the alphabet.
Abbreviate $C_{x \leqslant \top}$ as C.

Languages (easy!)

Consider $\top$ as a new letter in the alphabet.
Abbreviate $C_{x \leqslant \top}$ as C.

Define a reduction r as the homomorphic extension of:

$$\begin{cases} r(\top) = (\Sigma + \top)^\star & \\ r(a) = a & a \neq \top \end{cases}$$

Languages (easy!)

Consider $\top$ as a new letter in the alphabet.
Abbreviate $C_{x \leqslant \top}$ as C.

Define a reduction r as the homomorphic extension of:

$$\begin{cases} r(\top) = (\Sigma + \top)^\star & \\ r(a) = a & a \neq \top \end{cases}$$

We obtain:

$$\begin{tikzcd}
\mathrm{KA},\ x \leqslant \top \vdash e = f \arrow[rr, Leftrightarrow, "\text{reduction } r"] \arrow[dr, Rightarrow] & & C[e] = C[f] \\
& \mathrm{LANG} \models e = f \arrow[ur, Rightarrow] &
\end{tikzcd}$$

Languages (easy!)

Consider $\top$ as a new letter in the alphabet.
Abbreviate $C_{x \leqslant \top}$ as C.

Define a reduction r as the homomorphic extension of:

$$\begin{cases} r(\top) = (\Sigma + \top)^{\star} & \\ r(a) = a & a \neq \top \end{cases}$$

We obtain:

$$\begin{tikzcd}
\mathrm{KA},\ x \leqslant \top \vdash e = f \arrow[rr, Leftrightarrow, "\text{reduction } r"] \arrow[dr, Rightarrow, "\text{soundness}"'] & & C[e] = C[f] \\
& \mathrm{LANG} \models e = f \arrow[ur, Rightarrow] &
\end{tikzcd}$$

Languages (easy!)

Consider $\top$ as a new letter in the alphabet.
Abbreviate $C_{x\leqslant\top}$ as C.

Define a reduction r as the homomorphic extension of:

$$\begin{cases} r(\top) = (\Sigma + \top)^\star & \\ r(a) = a & a \neq \top \end{cases}$$

We obtain:

$$\begin{tikzcd}
\mathrm{KA},\ x \leqslant \top \vdash e = f \arrow[rr, Leftrightarrow, "\text{reduction } r"] \arrow[dr, Rightarrow, "\text{soundness}"'] & & C[e] = C[f] \\
& \mathrm{LANG} \models e = f \arrow[ur, Rightarrow, "C \text{ is a homomorphism}"'] &
\end{tikzcd}$$

Relations (harder!)

Abbreviate $C_{x \leqslant \top,\ x \leqslant x \top x}$ as D.

Relations (harder!)

Abbreviate $C_{x \leqslant \top,\ x \leqslant x \top x}$ as D.

We will obtain:

$$\begin{tikzcd}
\mathrm{KA},\ x \leqslant \top,\ x \leqslant x \top x \vdash e = f \arrow[r, Leftrightarrow] & D[e] = D[f] \arrow[d, Leftrightarrow] \\
\mathrm{REL} \models e = f \arrow[r, Leftrightarrow] & G(e) \lhd\!\rhd G(f)
\end{tikzcd}$$

Relations (harder!)

closed languages of words

Abbreviate $C_{x \leqslant \top,\ x \leqslant x \top x}$ as D.

We will obtain:

$$\begin{array}{rcl} \mathrm{KA},\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f & \Longleftrightarrow & D[e] = D[f] \\ & & \Updownarrow \\ \mathrm{REL} \models e = f & \Longleftrightarrow & G(e) \mathrel{\triangleleft\!\triangleright} G(f) \end{array}$$

Relations (harder!)

closed languages of words

Abbreviate $C_{x \leqslant \top,\ x \leqslant x \top x}$ as D.

We will obtain:

$$\mathrm{KA},\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f \Longleftrightarrow D[e] = D[f]$$

$$\Updownarrow$$

$$\mathrm{REL} \models e = f \Longleftrightarrow G(e) \lhd\!\rhd G(f)$$

languages of graphs, modulo homomorphisms

Relations (harder!)

closed languages of words

Abbreviate $C_{x \leqslant \top,\ x \leqslant x \top x}$ as D.

We will obtain:

$$\begin{array}{ccc}
\mathrm{KA},\ x \leqslant \top,\ x \leqslant x \top x \vdash e = f & \Longleftrightarrow & D[e] = D[f] \\
 & & \Updownarrow \\
\mathrm{REL} \models e = f & \underset{\text{[Brunet\&Pous '16]}}{\Longleftrightarrow} & G(e) \mathrel{\lhd\!\rhd} G(f)
\end{array}$$

languages of graphs, modulo homomorphisms

Relations (harder!)

closed languages of words

Abbreviate $C_{x\leqslant\top,\ x\leqslant x\top x}$ as D.

We will obtain:

$$\begin{array}{ccc}
\mathrm{KA},\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f & \stackrel{\text{a reduction}}{\Longleftrightarrow} & D[e] = D[f] \\
 & & \Updownarrow \\
\mathrm{REL} \models e = f & \underset{\text{[Brunet\&Pous '16]}}{\Longleftrightarrow} & G(e) \lhd\!\rhd G(f)
\end{array}$$

languages of graphs, modulo homomorphisms

Relations (harder!)

closed languages of words

Abbreviate $C_{x \leqslant \top,\ x \leqslant x \top x}$ as D.

We will obtain:

$$
\begin{array}{ccc}
\mathrm{KA},\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f & \overset{\text{a reduction}}{\Longleftrightarrow} & D[e] = D[f] \\
 & & \Updownarrow \text{a lemma} \\
\mathrm{REL} \models e = f & \underset{\text{[Brunet\&Pous '16]}}{\Longleftrightarrow} & G(e) \lhd\!\rhd G(f)
\end{array}
$$

languages of graphs, modulo homomorphisms

Third tool: graphs

Theorem

$\mathsf{REL} \models u \subseteq v$ *iff* $G(v) \lhd G(u)$

[Chandra & Merlin '77, Freyd & Scedrov '90, Andréka & Bredikhin '95]

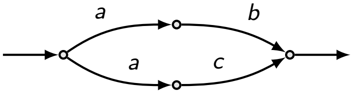
Third tool: graphs

Theorem

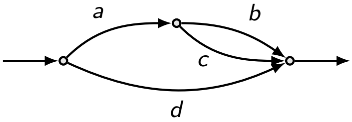
REL $\models u \subseteq v$ *iff* $G(v) \lhd G(u)$

[Chandra & Merlin '77, Freyd & Scedrov '90, Andréka & Bredikhin '95]

$a \cdot b \cap a \cdot c$



$a \cdot (b \cap c) \cap d$



Third tool: graphs

Theorem

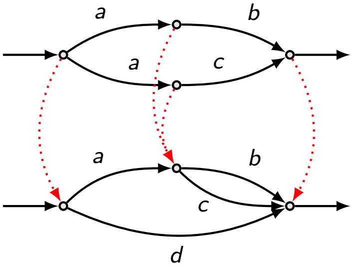
REL $\models u \subseteq v$ *iff* $G(v) \lhd G(u)$

[Chandra & Merlin '77, Freyd & Scedrov '90, Andréka & Bredikhin '95]

$a \cdot b \cap a \cdot c$

$|\cup$

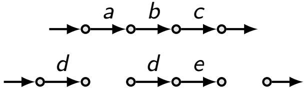
$a \cdot (b \cap c) \cap d$



Graphs of words with top

$G(abc)$

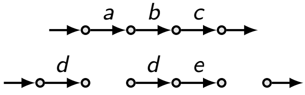
$G(d\top de\top)$



Graphs of words with top

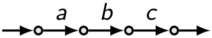
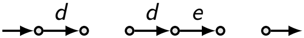
$G(abc)$

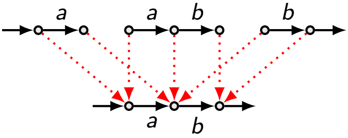
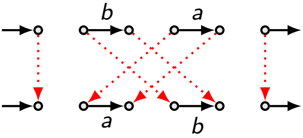
$G(d\top de\top)$



$ab \leqslant a\top ab\top b$	$\top a\top b\top \leqslant \top b\top a\top$

Graphs of words with top

$G(abc)$	
$G(d\top de\top)$	

$ab \leqslant a\top ab\top b$	$\top a\top b\top \leqslant \top b\top a\top$
	

A key lemma

Recall closures $C = C_{x \leqslant \top}$ and $D = C_{x \leqslant \top,\ x \leqslant x \top x}$.

Lemma

For all words with top u, v, the following are equivalent:

1. $u \leftsquigarrow^{\star}_{D} v$
2. $G(u) \lhd G(v)$
3. $u \in E(C(\{v\}))$ *where* $E(L) = \{u \mid \exists n,\ u(\top u)^n \in L\}$

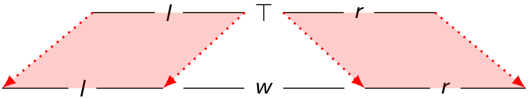
From rewriting to homomorphisms

Since $\lhd$ is a preorder, focus on single rewriting steps:

From rewriting to homomorphisms

Since $\lhd$ is a preorder, focus on single rewriting steps:

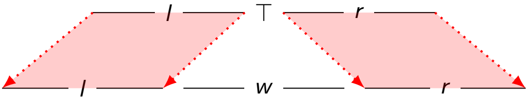
- $x \leqslant \top$, so that $lwr \leftsquigarrow l\top r$:



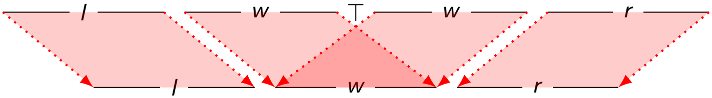
From rewriting to homomorphisms

Since $\lhd$ is a preorder, focus on single rewriting steps:

- $x \leqslant \top$, so that $lwr \leftsquigarrow l\top r$:

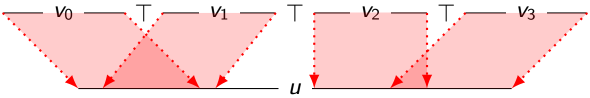


- $x \leqslant x\top x$, so that $lwr \leftsquigarrow lw\top wr$:



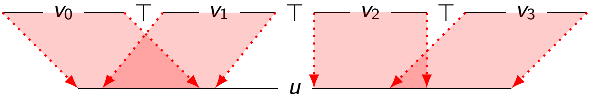
From homomorphisms to rewriting

Homomorphisms have the following shape:



From homomorphisms to rewriting

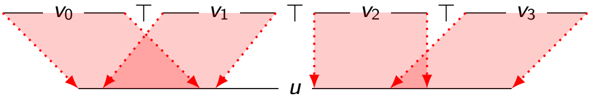
Homomorphisms have the following shape:



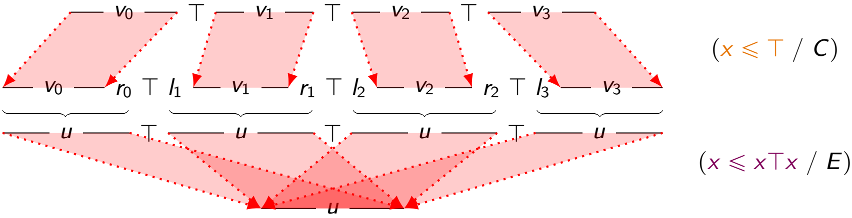
Thus $u = l_i v_i r_i$ for all i, with l_0 and r_3 empty.

From homomorphisms to rewriting

Homomorphisms have the following shape:



Thus $u = l_i v_i r_i$ for all i, with l_0 and r_3 empty.



A key lemma (proved)

Recall closures $C = C_{x \leqslant \top}$ and $D = C_{x \leqslant \top,\ x \leqslant x \top x}$.

Lemma

For all words with top u, v, the following are equivalent:

1. $u \leftsquigarrow^{\star} v$
2. $G(u) \lhd G(v)$
3. $u \in E(C(\{v\}))$ *where* $E(L) = \{u \mid \exists n,\ u(\top u)^n \in L\}$

A key lemma (proved)

Recall closures $C = C_{x \leqslant \top}$ and $D = C_{x \leqslant \top,\ x \leqslant x \top x}$.

Lemma

For all words with top u, v, the following are equivalent:

1. $u \leftsquigarrow^{\star} v$
2. $G(u) \lhd G(v)$
3. $u \in E(C(\{v\}))$ where $E(L) = \{u \mid \exists n,\ u(\top u)^n \in L\}$

Corollary

Previous vertical equivalence, and $D = E \circ C$.

Remaining reduction

We need to find a reduction t for D, i.e., such that

1. $D[e] = [t(e)]$, and
2. KA, $x \leqslant \top$, $x \leqslant x \top x \vdash e = t(e)$

Remaining reduction

We need to find a reduction t for D, i.e., such that

1. $D[e] = [t(e)]$, and
2. KA, $x \leqslant \top$, $x \leqslant x \top x \vdash e = t(e)$

This is harder than for C, because D is not a homomorphism.

Remaining reduction

We need to find a reduction t for D, i.e., such that

1. $D[e] = [t(e)]$, and
2. KA, $x \leqslant \top$, $x \leqslant x\top x \vdash e = t(e)$

This is harder than for C, because D is not a homomorphism.

However, since we already have reduction for C, and $D = E \circ C$, it suffices to find a reduction for E.

Remaining reduction

Recall that $E(L) = \{u \mid \exists n,\ u(\top u)^n \in L\}$.

We need to find a reduction s for E, i.e., such that

1. $E[e] = [s(e)]$, and
2. KA, $x \leqslant \top$, $x \leqslant x \top x \vdash e = s(e)$

Remaining reduction

Recall that $E(L) = \{u \mid \exists n,\ u(\top u)^n \in L\}$.

We need to find a reduction s for E, i.e., such that

1. $E[e] = [s(e)]$, and
2. KA, $x \leqslant \top$, $x \leqslant x \top x \vdash e = s(e)$

We use an automata construction for that.

An exercise in language theory

Given a language L, set

$$\sqrt[2]{L} = \{u \mid u^2 \in L\}$$

When L is regular, is $\sqrt[2]{L}$ regular?

An exercise in language theory

Given a language L, set

$$\sqrt[2]{L} = \{u \mid u^2 \in L\}$$
$$\sqrt{L} = \{u \mid \exists n \geqslant 1, u^n \in L\}$$

When L is regular, is $\sqrt[2]{L}$ regular?

Summary

Theorem

$$\mathrm{KA}, x \leqslant \top \vdash e = f \Leftrightarrow C[e] = C[f] \Leftrightarrow \mathrm{LANG} \models e = f$$

$$\mathrm{KA}, x \leqslant \top, x \leqslant x\top x \vdash e = f \Leftrightarrow D[e] = D[f] \Leftrightarrow \mathrm{REL} \models e = f$$

Using reductions, string rewriting, graphs, and monoids

Summary

Theorem

$$\mathrm{KA},\ x \leqslant \top \vdash e = f \Leftrightarrow C[e] = C[f] \Leftrightarrow \mathrm{LANG} \models e = f$$
$$\mathrm{KA},\ x \leqslant \top,\ x \leqslant x\top x \vdash e = f \Leftrightarrow D[e] = D[f] \Leftrightarrow \mathrm{REL} \models e = f$$

Using reductions, string rewriting, graphs, and monoids

More on this topic:

- PSPACE algorithm & KAT with top — arXiv/2304.07190
- general hypotheses & modular reductions — arXiv/2210.13020
- Rocq tactics — github/damien-pous/relation-algebra
- advanced decidability results: [Yoshiki Nakamura '17 and later]

Summary

Theorem

$$\mathrm{KA},\ x \leqslant \top \vdash e = f \Leftrightarrow C[e] = C[f] \Leftrightarrow \mathrm{LANG} \models e = f$$
$$\mathrm{KA},\ x \leqslant \top,\ x \leqslant x \top x \vdash e = f \Leftrightarrow D[e] = D[f] \Leftrightarrow \mathrm{REL} \models e = f$$

Using reductions, string rewriting, graphs, and monoids

More on this topic:

- PSPACE algorithm & KAT with top — `arXiv/2304.07190`
- general hypotheses & modular reductions — `arXiv/2210.13020`
- Rocq tactics — `github/damien-pous/relation-algebra`
- advanced decidability results: [Yoshiki Nakamura '17 and later]

Thanks again to Paul Brunet, Amina Doumane, Denis Kuperberg, Jurriaan Rot, and Jana Wagemaker