Monograph:
Foster, Simon David Kleene Algebra in Unifying Theories of Programming. Working Paper. (Unpublished)

Kleene Algebra in Unifying Theories of Programming

Simon Foster

April 5, 2018

Abstract

This development links Isabelle/UTP to the mechanised Kleene Algebra (KA) hiearchy for Isabelle/HOL. We substantiate the required KA laws, and provides a large body of additional theorems for alphabetised relations which are provided by the KA library. Additionally, we show how such theorems can be lifted to a subclass of UTP theories, provided certain conditions hold.

Contents

1 Kleene Algebra and UTP

theory *utp-kleene*
imports
KAT-and-DRA.KAT
UTP.utp
begin

This theory instantiates the Kleene Algebra [6] (KA) hierarchy, mechanised in Isabelle/HOL by Armstrong, Gomes, Struth et al [1, 4, 2]., for Isabelle/UTP alphabetised relations [3, 5]. Specifically, we substantiate the required dioid and KA laws in the type class hierarchy, which allows us to make use of all theorems proved in the former work. Moreover, we also prove an important result that a subclass of UTP theories, which we call "Kleene UTP theories", always form Kleene algebras. The proof of the latter is obtained by lifting laws from the KA hierarchy.

1.1 Syntax setup

It is necessary to replace parts of the KA syntax to ensure compatibility with UTP. We therefore delete various bits of notation, and hide some constants.

purge-notation *star* (-* [*101*] *100*)

recall-syntax

purge-notation *n-op* (*n* - [*90*] *91*)
purge-notation *ts-ord* (**infix** $\sqsubseteq$ *50*)

notation *n-op* ($\mathbf{n}$[-])
notation *t* ($\mathbf{n}^2$[-])
notation *ts-ord* (**infix** $\sqsubseteq_t$ *50*)

hide-const *t*

1.2 Kleene Algebra Instantiations

Next, import the laws of Kleene Algebra into the UTP relational calculus. We show that relations form a dioid and a Kleene algebra via two locales, the interpretation of which exports a large library of algebraic laws.

interpretation *urel-dioid*: *dioid*
where $plus = op \sqcap$ **and** $times = op\ ;;_h$ **and** $less\text{-}eq = less\text{-}eq$ **and** $less = less$
proof
fix $P\ Q\ R :: '\alpha\ hrel$
show $(P \sqcap Q) ;; R = P ;; R \sqcap Q ;; R$
by (*simp add*: *upred-semiring.distrib-right*)
show $(Q \sqsubseteq P) = (P \sqcap Q = Q)$
by (*simp add*: *semilattice-sup-class.le-iff-sup*)
show $(P < Q) = (Q \sqsubseteq P \wedge \neg P = Q)$
by (*simp add*: *less-le*)
show $P \sqcap P = P$
by *simp*
qed

interpretation *urel-ka*: *kleene-algebra*
where $plus = op \sqcap$ **and** $times = op\ ;;_h$ **and** $one = skip\text{-}r$ **and** $zero = false_h$ **and** $less\text{-}eq = less\text{-}eq$ **and** $less = less$ **and** $star = ustar$
proof
fix $P\ Q\ R :: '\alpha\ hrel$
show $II ;; P = P$ **by** *simp*
show $P ;; II = P$ **by** *simp*
show $false \sqcap P = P$ **by** *simp*
show $false ;; P = false$ **by** *simp*
show $P ;; false = false$ **by** *simp*
show $P^\star \sqsubseteq II \sqcap P ;; P^\star$
using *ustar-sub-unfoldl* **by** *blast*
show $Q \sqsubseteq R \sqcap P ;; Q \Longrightarrow Q \sqsubseteq P^\star ;; R$
by (*simp add*: *ustar-inductl*)
show $Q \sqsubseteq R \sqcap Q ;; P \Longrightarrow Q \sqsubseteq R ;; P^\star$
by (*simp add*: *ustar-inductr*)
qed

We also show that UTP relations form a Kleene Algebra with Tests [7, 4] (KAT).

interpretation *urel-kat*: *kat*
where $plus = op \sqcap$ **and** $times = op\ ;;_h$ **and** $one = skip\text{-}r$ **and** $zero = false_h$ **and** $less\text{-}eq = less\text{-}eq$ **and** $less = less$ **and** $star = ustar$ **and** $n\text{-}op = \lambda x.\ II \wedge (\neg x)$
by (*unfold-locales*, *rel-auto*+)

We can now access the laws of KA and KAT for UTP relations as below.

thm *urel-ka.star-inductr-var*
thm *urel-ka.star-trans*
thm *urel-ka.star-square*
thm *urel-ka.independence1*

1.3 Derived Laws

We prove that UTP assumptions are tests.

lemma *test-rassume* [*simp*]: *urel-kat.test* $[b]^{\top}$
  by (*simp add*: *urel-kat.test-def*, *rel-auto*)

The KAT laws can be used to prove results like the one below.

lemma *while-kat-form*:
  while b do P od $= ([b]^{\top} \;;;\; P)^{\star} \;;;\; [\neg\, b]^{\top}$ (**is** *?lhs* = *?rhs*)
proof −
  have *1*:$(II{::}'a\ hrel) \sqcap (II{::}'a\ hrel) \;;;\; [\neg\, b]^{\top} = II$
    by (*metis assume-true test-rassume urel-kat.test-absorb1*)
  have *?lhs* $= ([b]^{\top} \;;;\; P \sqcap [\neg\, b]^{\top} \;;;\; II)^{\star} \;;;\; [\neg\, b]^{\top}$
    by (*simp add*: *while-star-form rcond-rassume-expand*)
  also have ... $= (([b]^{\top} \;;;\; P)^{\star} \;;;\; [\neg\, b]^{\top\star})^{\star} \;;;\; [\neg\, b]^{\top}$
    by (*metis seqr-right-unit urel-ka.star-denest*)
  also have ... $= (([b]^{\top} \;;;\; P)^{\star} \;;;\; (II \sqcap [\neg\, b]^{\top})^{\star})^{\star} \;;;\; [\neg\, b]^{\top}$
    by (*metis urel-ka.star2*)
  also have ... $= (([b]^{\top} \;;;\; P)^{\star} \;;;\; (II)^{\star})^{\star} \;;;\; [\neg\, b]^{\top}$
    by (*metis 1 seqr-left-unit*)
  also have ... $= (([b]^{\top} \;;;\; P)^{\star})^{\star} \;;;\; [\neg\, b]^{\top}$
    by (*metis urel-ka.mult-oner urel-ka.star-one*)
  also have ... = *?rhs*
    by (*metis urel-ka.star-invol*)
  finally show *?thesis* **.**
qed

lemma *uplus-invol* [*simp*]: $(P^{+})^{+} = P^{+}$
  by (*metis RA1 uplus-def urel-ka.conway.dagger-trans-eq urel-ka.star-denest-var-2 urel-ka.star-invol*)

lemma *uplus-alt-def*: $P^{+} = P^{\star} \;;;\; P$
  by (*simp add*: *uplus-def urel-ka.star-slide-var*)

1.4 UTP Theories with Kleene Algebra

A Kleene UTP theory is continuous UTP theory with left and right units, and the top element as a left zero. The star in such a context has already been defined by lifting the relational Kleene star. Here, we use the KA theorems obtained above to provide corresponding theorems for a Kleene UTP theory.

locale *utp-theory-kleene* = *utp-theory-cont-unital-zerol*
begin

lemma *Star-def*: $P\bigstar = P^{\star} \;;;\; \mathcal{II}$
  by (*simp add*: *utp-star-def*)

lemma *Star-alt-def*:
  assumes P *is* $\mathcal{H}$
  shows $P\bigstar = \mathcal{II} \sqcap P^{+}$
proof −
  from *assms* **have** $P^{+} = P^{\star} \;;;\; P \;;;\; \mathcal{II}$
    by (*simp add*: *Unit-Right uplus-alt-def*)
  then show *?thesis*
    by (*simp add*: *RA1 utp-star-def*)
qed

lemma *Star-Healthy* [*closure*]:
  assumes P *is* $\mathcal{H}$
  shows $P\star$ *is* $\mathcal{H}$
  by (*simp add*: *assms closure Star-alt-def*)

lemma *Star-unfoldl*:
  $P\star \sqsubseteq \mathcal{II} \sqcap P \;;;\; P\star$
  by (*simp add*: *RA1 utp-star-def*)

lemma *Star-inductl*:
  assumes R *is* $\mathcal{H}$ $Q \sqsubseteq P \;;;\; Q \sqcap R$
  shows $Q \sqsubseteq P\star;;R$
proof −
  from *assms*(*2*) **have** $Q \sqsubseteq R$ $Q \sqsubseteq P \;;;\; Q$
    by *auto*
  thus *?thesis*
    by (*simp add*: *Unit-Left assms*(*1*) *upred-semiring.mult-assoc urel-ka.star-inductl utp-star-def*)
qed

lemma *Star-invol*:
  assumes P *is* $\mathcal{H}$
  shows $P\star\star = P\star$
  by (*metis* (*no-types*) *RA1 Unit-Left Unit-self assms urel-ka.star-invol urel-ka.star-sim3 utp-star-def*)

lemma *Star-test*:
  assumes P *is* $\mathcal{H}$ *utest* $\mathcal{T}$ P
  shows $P\star = \mathcal{II}$
  by (*metis utp-star-def Star-alt-def Unit-Right Unit-self assms semilattice-sup-class.sup.absorb1 semilattice-sup-class.sup-*
urel-ka.star-inductr-var-eq2 urel-ka.star-sim1 utest-def)

lemma *Star-lemma-1*:
  P *is* $\mathcal{H} \Longrightarrow \mathcal{II} \;;;\; P^\star \;;;\; \mathcal{II} = P^\star \;;;\; \mathcal{II}$
  by (*metis utp-star-def Star-Healthy Unit-Left*)

lemma *Star-lemma-2*:
  assumes P *is* $\mathcal{H}$ Q *is* $\mathcal{H}$
  shows $(P^\star \;;;\; Q^\star \;;;\; \mathcal{II})^\star \;;;\; \mathcal{II} = (P^\star \;;;\; Q^\star)^\star \;;;\; \mathcal{II}$
  by (*metis* (*no-types*) *assms RA1 Star-lemma-1 Unit-self urel-ka.star-sim3*)

lemma *Star-denest*:
  assumes P *is* $\mathcal{H}$ Q *is* $\mathcal{H}$
  shows $(P \sqcap Q)\star = (P\star \;;;\; Q\star)\star$
  by (*metis* (*no-types*, *lifting*) *RA1 utp-star-def Star-lemma-1 Star-lemma-2 assms urel-ka.star-denest*)

lemma *Star-denest-disj*:
  assumes P *is* $\mathcal{H}$ Q *is* $\mathcal{H}$
  shows $(P \vee Q)\star = (P\star \;;;\; Q\star)\star$
  by (*simp add*: *disj-upred-def Star-denest assms*)

lemma *Star-unfoldl-eq*:
  assumes P *is* $\mathcal{H}$
  shows $\mathcal{II} \sqcap P \;;;\; P\star = P\star$
  by (*simp add*: *RA1 utp-star-def*)

lemma *uplus-Star-def*:
assumes P *is* $\mathcal{H}$
shows $P^{+} = (P \;;;\; P\bigstar)$
by (*metis* (*full-types*) *RA1 utp-star-def Unit-Left Unit-Right assms uplus-def urel-ka.conway.dagger-slide*)

lemma *Star-trade-skip*:
P *is* $\mathcal{H} \Longrightarrow \mathcal{II} \;;;\; P^{\star} = P^{\star} \;;;\; \mathcal{II}$
by (*simp add*: *Unit-Left Unit-Right urel-ka.star-sim3*)

lemma *Star-slide*:
assumes P *is* $\mathcal{H}$
shows $(P \;;;\; P\bigstar) = (P\bigstar \;;;\; P)$ (**is** *?lhs* = *?rhs*)
proof −
have *?lhs* $= P \;;;\; P^{\star} \;;;\; \mathcal{II}$
by (*simp add*: *utp-star-def*)
also have ... $= P \;;;\; \mathcal{II} \;;;\; P^{\star}$
by (*simp add*: *Star-trade-skip assms*)
also have ... $= P \;;;\; P^{\star}$
by (*simp add*: *RA1 Unit-Right assms*)
also have ... $= P^{\star} \;;;\; P$
by (*simp add*: *urel-ka.star-slide-var*)
also have ... = *?rhs*
by (*metis RA1 utp-star-def Unit-Left assms*)
finally show *?thesis* **.**
qed

lemma *Star-unfoldr-eq*:
assumes P *is* $\mathcal{H}$
shows $\mathcal{II} \sqcap P\bigstar \;;;\; P = P\bigstar$
using *Star-slide Star-unfoldl-eq assms* **by** *auto*

lemma *Star-inductr*:
assumes P *is* $\mathcal{H}$ R *is* $\mathcal{H}$ $Q \sqsubseteq P \sqcap Q \;;;\; R$
shows $Q \sqsubseteq P;;R\bigstar$
by (*metis* (*full-types*) *RA1 Star-def Star-trade-skip Unit-Right assms urel-ka.star-inductr'*)

lemma *Star-Top*: $\top\bigstar = \mathcal{II}$
by (*simp add*: *Star-test top-healthy utest-Top*)

end

end

References

[1] A. Armstrong, V. Gomes, and G. Struth. Building program construction and verification tools from algebraic principles. *Formal Aspects of Computing*, 28(2):265–293, 2015.

[2] S. Foster, G. Struth, and T. Weber. Automated engineering of relational and algebraic methods in Isabelle/HOL. In *RAMICS*, LNCS 6663, pages 52–67. Springer, 2011.

[3] S. Foster, F. Zeyda, and J. Woodcock. Unifying heterogeneous state-spaces with lenses. In *Proc. 13th Intl. Conf. on Theoretical Aspects of Computing (ICTAC)*, volume 9965 of *LNCS*.

Springer, 2016.

[4] V. B. F. Gomes and G. Struth. Modal Kleene algebra applied to program correctness. In *Formal Methods*, volume 9995 of *LNCS*, pages 310–325. Springer, 2016.

[5] T. Hoare and J. He. *Unifying Theories of Programming*. Prentice-Hall, 1998.

[6] D. Kozen. On Kleene algebras and closed semirings. In *Proc. 15th Symp. on Mathematical Foundations of Computer Science (MFCS)*, volume 452 of *LNCS*, pages 26–47. Springer, 1990.

[7] D. Kozen. Kleene algebra with tests. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 19(3):427–443, 1997.