Automated Reasoning in Higher-Order Regular Algebra

Alasdair Armstrong and Georg Struth

Department of Computer Science
University of Sheffield, UK
{a.armstrong,g.struth}@dcs.shef.ac.uk

September 18, 2012

Overview

- Taken a large repository for first-order regular algebra in Isabelle/HOL
- Extended it towards higher order variants based on quantales
- Implemented substantial amounts of lattice theory to support this approach
- Developed useful theories and tools for working with regular algebra e.g.
    - Galois connections
    - Backhouse's fixpoint calculus
    - Order duality

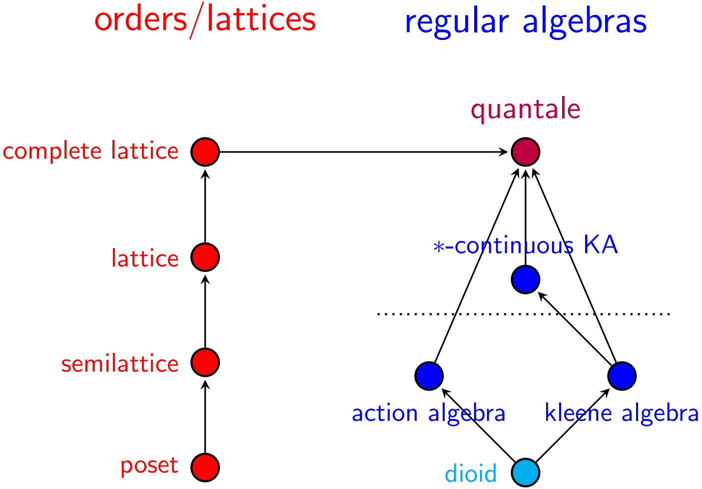
Overview

- Evaluated the effectiveness of ATP in this higher order setting
- Four case studies:
    1. Galois Connections
    2. Action Algebras
    3. Recursive Regular Equations
    4. Language Quantales

Overview

- Evaluated the effectiveness of ATP in this higher order setting
- Four case studies:
  1. **Galois Connections**
  2. **Action Algebras**
  3. Recursive Regular Equations
  4. Language Quantales

The Repository - An (Incomplete) Overview



Quantales

- A **quantale** is a structure $(Q, \leq, \cdot)$ such that $(Q, \leq)$ is a complete lattice, $\cdot$ is associative, and satisfying the infinite distributivity laws

$$x \left(\bigvee_{y \in Y} y \right) = \bigvee_{y \in Y} xy \qquad \text{and} \qquad \left(\bigvee_{y \in Y} y \right) x = \bigvee_{y \in Y} yx$$

- It is **unital** if $\cdot$ has an identity element 1
- The Kleene star can be defined as $x^* = \mu y.\ 1 + yx$
- Finite or infinite and infinite iteration,

$$x^\omega = \nu y.\ 1 + yx \qquad \text{and} \qquad x^\infty = \mu y.\ xy$$

Quantales - Without Explicit Carrier Sets

- The simplest way to define an algebraic structure in Isabelle is to use a class

class quantale = complete_lattice +
  fixes qmult :: "'a ⇒ 'a ⇒ 'a" (infixl "·" 80)
  assumes qmult_assoc: "$(x \cdot y) \cdot z = x \cdot (y \cdot z)$"
  and inf_distl: "$x \cdot \bigvee Y = \bigvee ((\lambda y.\ x{\cdot}y)\ `\ Y)$"
  and inf_distr: "$\bigvee Y \cdot x = \bigvee ((\lambda y.\ y{\cdot}x)\ `\ Y)$"

- Carrier set of the algebra is never explicitly mentioned—it's implicit in the type of qmult

Quantales - With Explicit Carrier Sets

- The alternative is to use locales and explicit carrier sets

locale quantale = **fixes** A (**structure**)
  assumes quantale_complete_lattice: "complete_lattice A"
  and mult_type: "op $\cdot \in$ carrier A $\to$ carrier A $\to$ carrier A"
  and mult_assoc: "$[\![x \in \text{carrier } A;\ y \in \text{carrier } A;\ z \in \text{carrier } A]\!]$
    $\Longrightarrow (x \cdot y) \cdot z = x \cdot (y \cdot z)$"
  and inf_distl: "$[\![x \in \text{carrier } A;\ Y \subseteq \text{carrier } A]\!]$
    $\Longrightarrow x \cdot \bigvee Y = \bigvee ((\lambda y.\ x{\cdot}y) \text{ ` } Y)$"
  and inf_distr: "$[\![x \in \text{carrier } A;\ Y \subseteq \text{carrier } A]\!]$
    $\Longrightarrow \bigvee Y \cdot x = \bigvee ((\lambda y.\ y{\cdot}x) \text{ ` } Y)$"

- Now we can use any arbitrary Isabelle set as our carrier set

Fixpoints

- Many useful fixpoint theorems in the repository
  - Knaster-Tarski theorem
  - Kleene's fixed point theorem
  - Fixpoint Fusion
- Rules from fixpoint calculus implemented, and useful for reasoning with fixed points
- Iteration operators in quantales defined as fixed points

definition is_lfp :: "('a, 'b) ord_scheme ⇒ 'a ⇒ ('a ⇒ 'a) ⇒ bool" **where**
"is_lfp A x f ≡ f x = x ∧ (∀y∈carrier A. f y = y ⟶ x $\leq_A$ y)"

definition least_fixpoint :: "('a, 'b) ord_scheme ⇒ ('a ⇒ 'a) ⇒ 'a"
("μ_" [0,1000] 100) **where**
"least_fixpoint A f ≡ THE x. is_lfp A x f"

Knaster-Tarski (for least fixed points)

```
theorem knaster_tarski_lpp:
  assumes cl_A: "complete_lattice A" and f_closed: "f ∈ carrier A → carrier A"
  and f_iso: "isotone A A f"
  shows "∃!x. is_lpp A x f"
proof
  let ?H = "{u. f u ≤_A u ∧ u ∈ carrier A}"
  let ?a = "⋀_A ?H"

  have H_carrier: "?H ⊆ carrier A" by (metis (lifting) mem_Collect_eq subsetI)
  hence a_carrier: "?a ∈ carrier A"
    by (smt order.glb_closed complete_meet_semilattice.is_glb_glb . . . )
```

Knaster-Tarski (for least fixed points)

```
have "is_pre_fp A ?a f"
proof -
  have "∀x∈?H. ?a ≤_A x" by (smt H_carrier ...)
  hence "∀x∈?H. f ?a ≤_A f x"
    by (safe, rule_tac ?f = f in use_iso1, metis f_iso, metis a_carrier, auto)
  hence "∀x∈?H. f ?a ≤_A x" by (smt CollectD a_carrier cl_A ...)
  hence "f ?a ≤_A ?a" by (smt complete_meet_semilattice.glb_greatest ...)
  thus ?thesis by (smt a_carrier cl_A cl_to_order f_closed is_pre_fp_def)
qed
moreover show "∧x. is_lpp A x f ⟹ x = ?a"
  by (smt H_carrier calculation cl_A cl_to_cms ...)
ultimately show "is_lpp A ?a f"
  by (smt H_carrier cl_A cl_to_cms complete_meet_semilattice.glb_least ...)
qed
```

Knaster-Tarski (for greatest fixed points)

- Dual theorems can easily be proved
- The $\sharp$ operator maps an order to it's dual
- We state the dual of the theorem we want to prove using $\sharp$
- The simplifier can then simplify away all the instances of $\sharp$, proving the theorem we want

```
theorem knaster_tarski_gpp:
  assumes cl_A: "complete_lattice A" and f_closed: "f ∈ carrier A → carrier A"
  and f_iso: "isotone A A f"
  shows "∃!x. is_gpp A x f"
proof -
  have dual:
    "⟦complete_lattice (A♯); f ∈ carrier (A♯) → carrier (A♯); isotone (A♯) (A♯) f⟧
    ⟹ ∃!x. is_lpp (A♯) x f"
    by (smt knaster_tarski_lpp)
  thus ?thesis by (simp, metis cl_A f_closed f_iso)
qed
```

Quantales - Example Proof

We can show that x^* is equivalent to $\bigvee_{n \in \mathbb{N}} x^n$ using

Kleene's fixed point theorem

For any Scott-continuous function f over a complete partial order, the least fixed point of f is also the least upper bound of the ascending Kleene chain of f

$$\mu(f) = \bigvee_{n \in \mathbb{N}} f^n(\bot)$$

This shows us that

$$x^* = 1 + (1 + x) + (1 + x + x^2) + (1 + x + x^2 + x^3) + \dots$$

Quantales - Example Proof

- We can then use the rule that in any complete lattice $(A, \leq)$,

$$\bigvee \left\{ \bigvee Y \middle| Y \in X \right\} = \bigvee \left(\bigcup X \right) \quad \text{where } X \subseteq \mathcal{P}(A)$$

  to complete to proof $\square$
- The repository allows this reasoning to be used within Isabelle.
- Availability of theorems from fixpoint calculus and lattice theory makes reasoning in regular algebra much easier

```
lemma star_power: assumes xc: "x ∈ carrier A" shows "x* = Σ (powers x)"
proof -
  let ?STAR_FUN = "λy. 1 + x·y"

  have star_chain: "μ_A ?STAR_FUN = Σ (carrier (kleene_chain A ?STAR_FUN))"
  proof (rule kleene_fixed_point, unfold_locales)
    show "?STAR_FUN ∈ carrier A → carrier A"
      by (smt ftype_pred one_closed mult_closed join_closed xc)
  next
    show "isotone A A ?STAR_FUN"
      by (simp add: isotone_def, safe, metis quantale_order, smt ...)
  next
    fix D assume "D ⊆ carrier A"
    and "directed (|carrier = D, le = op ≤, ... = ord.more A|)"
    thus "1 + x · Σ D = Σ ((λy. 1 + x · y) ` D)"
      by (metis assms star_scott_continuous)
  qed
```

```
have "μ_A ?STAR_FUN = Σ {z. ∃i. z = Σ (powersUpTo i x)}"
  by (simp add: star_chain kleene_chain_def iter_powersUpTo)
moreover have "... = Σ (Σ ` {z. ∃i. z = powersUpTo i x})"
  by (rule_tac ?f = "λY. Σ Y" in arg_cong, safe, auto+)
moreover have "... = Σ (⋃ {z. ∃i. z = powersUpTo i x})"
  by (rule lub_denest, safe, auto, simp add: powersUpTo_def, safe, metis . . . )
moreover have "... = Σ (powers x)"
  apply (rule_tac ?f = "λY. Σ Y" in arg_cong, safe, auto+)
  apply (simp_all add: powersUpTo_def powers_def, metis)
  by (metis (lifting, full_types) le_add2 mem_Collect_eq)
ultimately show ?thesis
  by (metis star_def)
qed
```

Case Study 1 - Galois Connections

- A **Galois connection** between two posets $(A, \leq_A)$ and $(B, \leq_B)$ is a pair of functions $f : A \to B$ and $g : B \to A$ such that forall $x \in A$ and $y \in B$

$$f(x) \leq_A y \longleftrightarrow x \leq_A g(y)$$

- Theorems for free! For example, $f : A \to B$ is the lower adjoint in a Galois connection between two complete lattices iff

$$\bigvee_{x \in X} f(x) = f(\bigvee_{x \in X} x)$$

Galois Connections in Isabelle

locale galois_connection =
  fixes orderA :: "('a, 'c) ord_scheme" ("α")
  and orderB :: "('b, 'd) ord_scheme" ("β")
  and lower :: "'a → 'b" ("π^*")
  and upper :: "'b → 'a" ("π_*")
  assumes is_order_A: "order α"
  and is_order_B: "order β"
  and lower_closure: "$\pi^* \in$ carrier $\alpha \to$ carrier β"
  and upper_closure: "$\pi_* \in$ carrier $\beta \to$ carrier α"
  and galois_property:
    "$[\![\pi^*$ x $\in$ carrier β; x $\in$ carrier α; y $\in$ carrier β; π_* y $\in$ carrier $\alpha]\!]$
    $\Longrightarrow \pi^*$ x $\leq_\beta$ y $\longleftrightarrow$ x $\leq_\alpha$ π_* y"

Galois Connections - ATP Support

- Multiple orders with carrier sets necessary for many interesting applications
- Galois connections between two endofunctions without carrier sets can easily be reasoned about with ATP
- Without carrier sets proofs become much more manual

Case Study 2 - Action Algebras

- Kleene algebra expanded with two residuation operations

$$(A, +, 0, \cdot, 1, \leftarrow, \rightarrow, ^*)$$

- Axioms:

$$xy \leq z \Leftrightarrow x \leq z \leftarrow y \quad \text{and} \quad xy \leq z \Leftrightarrow y \leq x \rightarrow z$$

$$1 + x^*x^* + x \leq x^* \quad \text{and} \quad 1 + yy + x \leq y \Rightarrow x^* \leq y$$

- Properties of residuation can be instantiated from Galois connections
- First-order regular algebra — trivial for ATP systems

Quantales - Galois Connections

- Recall that f is the lower adjoint in a Galois connection iff

$$\bigvee_{x \in X} f(x) = f(\bigvee_{x \in X} x)$$

- This immediately implies that $(x\cdot)$ has an upper adjoint
- Preimplication/residuation operator $(x \rightarrow)$
- $(\cdot x)$ also has an upper adjoint $(\leftarrow x)$
- Trivial to show that $(Q, +, 0, \cdot, 1, \leftarrow, \rightarrow, ^*)$ is an action algebra
- Theorems from action algebra then availabe in quantales

Conclusion

- Heirachy of lattices and regular algebras formalised in Isabelle
- Additional theories such as fixpoints and Galois connections provide powerful proof support
- Automated tools still useful in a Higher-order setting
- Usable for many applications
- Available online:
- `https://github.com/Alasdair/IsabelleAlgebra`