formalizing approximation fixpoint theory in coq

luís cruz-filipe[1] bart bogaerts[2]

[1] department of mathematics and computer science
university of southern denmark

[2] department of computer science
vrije universiteit brussel

days in logic
july 1st, 2022

why formalizations

- 2001–2004: C-CoRN
  with H. Geuvers, B. Spitters, F. Wiedijk, . . .
- 2014–2015: Sorting networks
  with P. Schneider-Kamp
- 2016–2018: SAT solving
  with J. Marques-Silva, A. Rebola-Pardo, P. Schneider-Kamp
- currently: several ongoing projects

why approximation fixpoint theory

2013–2018: active integrity constraints
(with G. Gaspar and I. Nunes)

- rules for repairing database inconsistencies
- no operational semantics
- no "good" semantics

why approximation fixpoint theory

2013–2018: active integrity constraints
(with G. Gaspar and I. Nunes)

- rules for repairing database inconsistencies
- no operational semantics
- no "good" semantics

$\rightsquigarrow$ until: approximation fixpoint theory

approximation fixpoint theory

what

a framework for studying fixpoints

- of operators over complete lattices
- of approximators to these operators

approximation fixpoint theory

what

a framework for studying fixpoints

- of operators over complete lattices
- of approximators to these operators

why

unifying theory for many different constructions in (non-monotonic) logics

approximation fixpoint theory

what

a framework for studying fixpoints

- of operators over complete lattices
- of approximators to these operators

why

unifying theory for many different constructions in (non-monotonic) logics

how

heavy use of transfinite sequences and transfinite induction

the classical example

knaster-tarski theorem

every monotonic operator over a complete lattice has a least fixpoint, which can be obtained by transfinite iteration

- classical semantics of logic programming &c

the classical example

knaster-tarski theorem

every monotonic operator over a complete lattice has a least fixpoint, which can be obtained by transfinite iteration

- classical semantics of logic programming &c

less classical examples

- extensions (models) of reiter's default logic
- answer sets semantics for logic programming
- well-founded semantics for logic programming

the generalization

approximation fixpoint theory

- defines operators and approximators over complete lattices
- defines their fixpoints (abstractly)
- captures and generalizes known semantics

example domains

- logic programming (of course)
- autoepistemic logics
- default logics
- argumentation theory
- description logics
- active integrity constraints

grounded fixpoints

grounded points

a point x in a lattice L is *grounded* for $O : L \to L$ if: whenever $O(v \wedge x) \leq v$, it holds that $x \leq v$

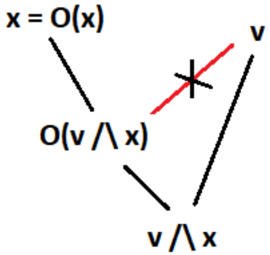
grounded fixpoints, intuitively

if we remove "a part" of x, then applying O will readd a portion of the removed part

grounded fixpoints

grounded points

a point x in a lattice L is *grounded* for $O : L \to L$ if: whenever $O(v \wedge x) \leq v$, it holds that $x \leq v$



grounded fixpoints

grounded points

a point x in a lattice L is *grounded* for $O : L \to L$ if: whenever $O(v \wedge x) \leq v$, it holds that $x \leq v$

grounded fixpoints, intuitively

if we remove "a part" of x, then applying O will readd a portion of the removed part

$\rightsquigarrow$ grounded fixpoints capture "non-circularity" of models

approximator lattice

let L be a complete lattice

bilattice L^2

intuitively: points correspond to “intervals”

- consistent elements: (x, y) with $x \leq y$
- exact elements: (x, x)
- precision ordering: $(x, y) \leq_p (u, v)$ iff $x \leq u$ and $v \leq y$ ((u, v) is “more precise” than (x, y))

L^2 is a complete lattice

approximators

let $O : L \to L$ be an operator

approximator of O

$A : L^2 \to L^2$ is an approximator of O if A is monotonic and $A(x, x) = (O(x), O(x))$

- typically *symmetric*: $A(x, y)_1 = A(y, x)_2$
- map consistent elements to consistent elements

some fixpoints of interest

assume A is an approximator of O

- the A-kripke-kleene fixpoint is the lfp of A (it approximates all fixpoints of O)
- a partial-A-stable fixpoint is a pair (x, y) such that $x = \mathrm{lfp} A(\cdot, y)_1$ and $y = \mathrm{lfp} A(x, \cdot)_2$
- the A-well-founded fixpoint is the minimal partial-A-stable fixpoint
- an A-stable fixpoint of O is a fixpoint x of O s.t. (x, x) is a minimal partial-A-stable fixpoint

the coq formalization

design decisions

- constructive (as far as possible)
- follow the mathematical development closely

main challenges

adapt proofs relying on some classical decidability properties

ordinals

type of unbounded sets of ordinals

an ordinal consists of:

- a type T
- binary relations $=$ and $<$ on T
- an element $0 : T$
- a function $S : T \to T$
- and a gazillion axioms

$\rightsquigarrow$ S makes the set unbounded

ordinals

type of unbounded sets of ordinals

an ordinal consists of:

- a type T
- binary relations $=$ and $<$ on T
- an element $0 : T$
- a function $S : T \to T$
- and a gazillion axioms

$\rightsquigarrow$ S makes the set unbounded

- examples: ω, ω^ω, towers of $\omega\ldots$

complete lattices

very standard

a lattice consists of:

- a type C
- binary relations $=$ and $\leq$ on C
- a function $\mathsf{lub} : (C \to \mathsf{Prop}) \to C$
- and a few axioms

- example: powersets
- constructions: bilattice, dual

complete lattices

very standard

a lattice consists of:

- a type C
- binary relations $=$ and $\leq$ on C
- a function $\mathsf{lub} : (C \to \mathsf{Prop}) \to C$
- and a few axioms

fixpoints

the knaster-tarksi theorem – existence of fixpoints and operational characterizations using chains and O-inductions

complete lattices

very standard

a lattice consists of:

- a type C
- binary relations $=$ and $\leq$ on C
- a function $\mathsf{lub} : (C \to \mathsf{Prop}) \to C$
- and a few axioms

approximators

definitions and main properties

what's next?

ongoing

formalizing a complete example

- (propositional) logic programming
- syntax
- traditional semantics (directly)
- aft traditional (via approximators)
- proofs of correspondence

thank you!