First Order Logic With Fixed-points and Cyclic Proofs

Charlie Murphy

April 26, 2024

Overview

- First Order Logic with Fixed-Points
  - First Order Logic
  - Fixed-Points
  - First Order Logic with Fixed-Points
- Proof Systems
  - Inference Rules / Non-Cyclic Case
  - Cyclic Proof Systems
  - Property Directed Reachability as Cyclic Proof Search

First Order Logic

First Order Logic

- Allows one to unambiguously formalize statements:
  - Every person has a mother: $\forall p.\, \mathrm{person}(p) \Rightarrow \exists m.\, \mathrm{motherOf}(m, p)$

- The language of first-order formulas over signature Σ:

$$t ::= x \mid \mathrm{f}(\mathrm{t}_1, \ldots, \mathrm{t}_{ar(f)})$$
$$\phi ::= X\left(t_1, \ldots, t_{ar(X)}\right) \mid p\left(t_1, \ldots, t_{ar(p)}\right) \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \forall\, x : s.\, \phi$$

  - All other logical connectives are definable:
    - E.g., $\phi_1 \wedge \phi_2 \stackrel{\text{def}}{=} \neg(\neg\phi_1 \vee \neg\phi_2)$

First Order Theories (over signature Σ)

- A first order theory is a set of first-order formulas:
  - E.g., Peano Arithmetic, Linear Real/Rational Arithmetic, Linear Integer Arithmetic, Theory of Arrays, Theory of Algebraic Datatypes, etc.

- A first order theory structure $\mathcal{T} \stackrel{\text{def}}{=} \langle D, I \rangle$ consists of:
  - A universe of objects D (D_S is the universe of objects of sort S)
  - An interpretation function I for predicate and function symbols in Σ

First Order Satisfiability

Let $\mathcal{T} \stackrel{\text{def}}{=} \langle D, I \rangle$ be a first order structure over signature Σ and M a model that maps variables to elements of universe D.

$$[\![x]\!](M) \stackrel{\text{def}}{=} M(x) \qquad [\![f(\bar{t})]\!](M) \stackrel{\text{def}}{=} I(f)([\![\bar{t}]\!](M))$$

$$[\![\neg\phi]\!](M) \stackrel{\text{def}}{=} \neg[\![\phi]\!](\mathrm{M}) \qquad [\![\phi_1 \vee \phi_2]\!](M) \stackrel{\text{def}}{=} [\![\phi_1]\!](M) \vee [\![\phi_2]\!](M)$$

$$[\![X(\bar{t})]\!](M) \stackrel{\text{def}}{=} M(X)([\![\bar{t}]\!](M)) \qquad [\![p(\bar{t})]\!](M) \stackrel{\text{def}}{=} I(p)([\![\bar{t}]\!](M))$$

$$[\![\forall x : S.\, \phi]\!](M) \stackrel{\text{def}}{=} \bigvee_{v \in D_S} [\![\phi]\!](M[x \mapsto v])$$

First Order Satisfiability

Let $\mathcal{T} \stackrel{\text{def}}{=} \langle D, I \rangle$ be a first order structure over signature Σ,
M a model that maps variables to elements of universe D, and
ϕ a first-order formula over signature Σ.

M satisfies ϕ ($M \vDash \phi$) if and only if $[\![\phi]\!](M') = \text{true}$ for all extensions M' of M

ϕ is valid ($\vDash \phi$) if and only if $\emptyset \vDash \phi$

Fixed-points

Fixed-points

- For any sort S and function $f : S \to S$ a fixed-point of f is any point $x \in S$ such that $x = f(x)$. For example:
  - $f(x) = 2x$ has one fixed-point 0
  - $f(x) = x^3$ has three fixed-points -1, 0, and 1.
  - $f(x) = x$ has infinitely many fixed-points
  - $f(x) = x + 1$ has zero fixed-points

Occurrences of Fixed-points

- Program Semantics (e.g., while loops, recursive functions)
- Algebraic Data Types
- Induction and Co-Induction
- Abstract Interpretation
- Invariant Generation
- Model Checking

Greatest and Least Fixed-points

Let $\langle L, \leq \rangle$ be a complete lattice and $F : L \rightarrow L$ a monotonic function on L

F has a greatest fixed-point x

for all fixed-points y, x is larger than y (i.e., $y \leq x$)

F has a least fixed-point x

for all fixed-points y, x is less than y (i.e., $x \leq y$)

[Knaster Tarski Fixed-point Theorem]

Greatest and Least Fixed-points

Let $\langle L, \leq \rangle$ be a complete lattice and $F : L \to L$ a monotonic function on L

The greatest fixed-point of F is $\nu x. F(x) \stackrel{\text{def}}{=} F^{\tau}(\top)$
(for some sufficiently large ordinal τ)

The greatest fixed-point of F is $\nu x. F(x) \stackrel{\text{def}}{=} F^{\tau}(\bot)$
(for some sufficiently large ordinal τ)

[Knaster Tarski Fixed-point Theorem]

First Order Logic with Fixed-points

Fixed-points in First Order Logic

- Typically fixed-points occur either implicitly or explicitly when using uninterpreted relations
  - Least Fixed-Points:
    - Constraint Logic Programming (CLP)
      - E.g., for solving Constraint Satisfaction Problems
    - Constrained Horn Clauses (CHCs)
      - $\forall \overline{x_0}, \dots, \overline{x_n}. X_0(\overline{x_0}) \Leftarrow X_1(\overline{x_1}) \land \cdots \land X_n(\overline{x_n}) \land \phi(\overline{x_0}, \dots, \overline{x_n})$
  - Greatest Fixed-Points:
    - Constraint Logic Programming
      - Finding most general solution
    - Co-Constrained Horn Clauses (coCHCs)
      - $\forall \overline{x_0}, \dots, \overline{x_n}. X_0(\overline{x_0}) \Rightarrow X_1(\overline{x_1}) \lor \cdots \lor X_n(\overline{x_n}) \lor \phi(\overline{x_0}, \dots, \overline{x_n})$

CHCs as Least Fixed-Points

```
0:   While 0 < x
1:     x--;
2:     y++;
```

$$\frac{0 \geq x \quad x = x' \quad y = y'}{sem^0(x, y, x', y')}$$

$$\frac{0 < x \quad sem^{1,2}(x, y, x'', y'') \quad sem^0(x'', y'', x', y')}{sem^0(x, y, x', y')}$$

$$\frac{x' = x - 1 \quad y = y'}{sem^1(x, y, x', y')}$$

$$\frac{sem^1(x, y, x'', y'') \quad sem^2(x'', y'', x', y')}{sem^{1,2}(x, y, x', y')}$$

$$\frac{x' = x \quad y' = y + 1}{sem^1(x, y, x', y')}$$

muCLP Calculus

- muCLP extends Constraint Logic Programming (CLP)
  - Adds explicit use of least and greatest fixed-point operators to define the meaning of uninterpreted relations
  - Generalizes both CHCs and coCHCs

Unno et. al. “Modular Primal-Dual Fixpoint Logic Solving for Temporal Verification.” POPL, 2023.

muCLP Calculus

A muCLP formula for theory $\mathcal{T}$ takes the following form

$$
\begin{aligned}
&\phi_0 \text{ s.t.} \\
&\ \mathrm{X}_1(\overline{x_1}) =_{\alpha_1} \phi_1; \\
&\ \ldots; \\
&\ X_n(\overline{x_n}) =_{\alpha_n} \phi_n
\end{aligned}
$$

Where each X_i is a predicate variable, $\overline{x_i}$ is a sequence of term variables, ϕ_i is a first-order formula that may include positive occurrences of the predicate variables X_1 through X_n and the term variables $\overline{x_i}$, and each α_i is either μ representing a least fixed-point or ν representing a greatest fixed-point.

muCLP Example

```
0:  While 0 < x
1:      x--;
2:      y++;
```

$$\forall x, y, x', y'.\ sem^0(x, y, x', y') \Downarrow sem^0(y, x, x', y')$$

Semantics

$$\forall x, y, x', y'. \overline{sem^0}(x, y, x', y') \Rightarrow sem^0(y, x, x', y')\ s.t.$$

$$sem^0(x, y, x', y') =_\mu \vee \begin{pmatrix} (0 \geq x \wedge x' = x \wedge y' = y) \\ 0 < x \wedge \exists x'', y''. sem^{1,2}(x, y, x'', y'') \wedge sem^0(x'', y'', x', y') \end{pmatrix};$$

$$sem^{1,2}(x, y, x', y') =_\mu \exists x'', y''. sem^1(x, y, x'', y'') \wedge sem^2(x'', y'', x', y');$$

$$sem^1(x, y, x', y') =_\mu x' = x - 1 \wedge y' = y;$$

$$sem^2(x, y, x', y') =_\mu x' = x \wedge y' = y + 1;$$

$$\overline{sem^0}(x, y, x', y') =_\nu \wedge \begin{pmatrix} (0 < x \vee x' \neq x \vee y' \neq y) \\ 0 \geq x \vee \forall x'', y''. \overline{sem^{1,2}}(x, y, x'', y'') \vee \overline{sem^0}(x'', y'', x', y') \end{pmatrix};$$

$$\overline{sem^{1,2}}(x, y, x', y') =_\nu \forall x'', y''. \overline{sem^1}(x, y, x'', y'') \vee \overline{sem^2}(x'', y'', x', y');$$

$$\overline{sem^1}(x, y, x', y') =_\nu x' \neq x - 1 \vee y' \neq y;$$

$$\overline{sem^2}(x, y, x', y') =_\nu x' \neq x \vee y' \neq y + 1$$

Dual Semantics

muCLP Satisfiability

A muCLP formula $\mathcal{P} \stackrel{\text{def}}{=} \phi \ s.t.\ P$ is satisfiable if and only if $\llbracket P \rrbracket(\emptyset) \vDash \phi$, where $\llbracket P \rrbracket(\emptyset)$ maps each predicate variable defined in P to its fixed-point.

$$X =_\nu X \wedge Y; Y =_\mu X \vee Y$$

$$\llbracket X =_\nu X \wedge Y; Y =_\mu X \vee Y \rrbracket \stackrel{\text{def}}{=} \nu X. X \wedge (\mu Y. X \vee Y) \stackrel{\text{def}}{=} \{X \mapsto \top, Y \mapsto \top\}$$

$$Y =_\mu X \vee Y;\ X =_\nu X \wedge Y$$

$$\llbracket Y =_\mu X \vee Y;\ X =_\nu X \wedge Y \rrbracket \stackrel{\text{def}}{=} \mu Y. (\nu X. X \wedge Y) \vee Y \stackrel{\text{def}}{=} \{X \mapsto \bot, Y \mapsto \bot\}$$

muCLP Example

```
0:   While 0 < x
1:     x--;
2:     y++;
```

$$\forall x,y,x',y'.\ sem^0(x,y,x',y') \Downarrow sem^0(y,x,x',y')$$

$$\forall x,y,x',y'.\overline{sem^0}(x,y,x',y') \Rightarrow sem^0(y,x,x',y')\ s.t.$$

$$sem^0(x,y,x',y') =_\mu \vee \begin{pmatrix} (0 \geq x \wedge x' = x \wedge y' = y) \\ 0 < x \wedge \exists x'',y''.\, sem^{1,2}(x,y,x'',y'') \wedge sem^0(x'',y'',x',y') \end{pmatrix};$$

$$sem^{1,2}(x,y,x',y') =_\mu \exists x'',y''.\, sem^1(x,y,x'',y'') \wedge sem^2(x'',y'',x',y');$$

$$sem^1(x,y,x',y') =_\mu\ x' = x - 1 \wedge y' = y;$$

$$sem^2(x,y,x',y') =_\mu\ x' = x \wedge y' = y + 1;$$

$$\overline{sem^0}(x,y,x',y') =_\nu \wedge \begin{pmatrix} (0 < x \vee x' \neq x \vee y' \neq y) \\ 0 \geq x \vee \forall x'',y''.\, \overline{sem^{1,2}}(x,y,x'',y'') \vee \overline{sem^0}(x'',y'',x',y') \end{pmatrix};$$

$$\overline{sem^{1,2}}(x,y,x',y') =_\nu \forall x'',y''.\, \overline{sem^1}(x,y,x'',y'') \vee \overline{sem^2}(x'',y'',x',y');$$

$$\overline{sem^1}(x,y,x',y') =_\nu\ x' \neq x - 1 \vee y' \neq y;$$

$$\overline{sem^2}(x,y,x',y') =_\nu\ x' \neq x \vee y' \neq y + 1$$

$$sem^0 \stackrel{\text{def}}{=} \lambda x,y,x',y'.(0 \geq x \wedge x' = x \wedge y' = y) \vee (0 < x \wedge\ 0 = x' \wedge\ y' = y + x)$$

$$\overline{sem^0} \stackrel{\text{def}}{=} \lambda x,y,x',y'.(0 < x \vee x' \neq x \vee y' \neq y) \wedge (0 \geq x \vee\ 0 \neq x' \vee\ y' = y + x)$$

Proof Systems

Proof Systems

- A proof system consists of
  - A set of axioms (or schematic axioms)
  - Rules of Inference
- Example Proof Systems:
  - Resolution (e.g., for formulas in conjunctive normal form)
  - Hilbert Proof System (e.g., axioms and modus ponens)
  - Sequent Calculus (e.g., for propositional and first-order logic)

Sequent Calculus (Propositional Logic)

$$\frac{}{\Gamma, A \vdash \Delta, \mathrm{A}}\text{Atom}$$

$$\frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta}\wedge L \qquad \frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta}\vee L \qquad \frac{\Gamma \vdash \Delta, \mathrm{A} \quad \Gamma, B \vdash \Delta}{\Gamma, A \to B \vdash \Delta}\to L \qquad \frac{\Gamma \vdash \Delta, \mathrm{A}}{\Gamma, \neg A \vdash \Delta}\neg L$$

$$\frac{\Gamma \vdash \Delta, \mathrm{A} \quad \Gamma \vdash \Delta, \mathrm{B}}{\Gamma \vdash \Delta, \mathrm{A} \wedge B}\wedge R \qquad \frac{\Gamma \vdash \Delta, \mathrm{A}, \mathrm{B}}{\Gamma \vdash \Delta, \mathrm{A} \vee B}\vee R \qquad \frac{\Gamma, A \vdash \Delta, \mathrm{B}}{\Gamma, A \to B \vdash \Delta}\to R \qquad \frac{\Gamma, \mathrm{A} \vdash \Delta}{\Gamma \vdash \Delta, \neg A}\neg R$$

Sequent Calculus Example Proof

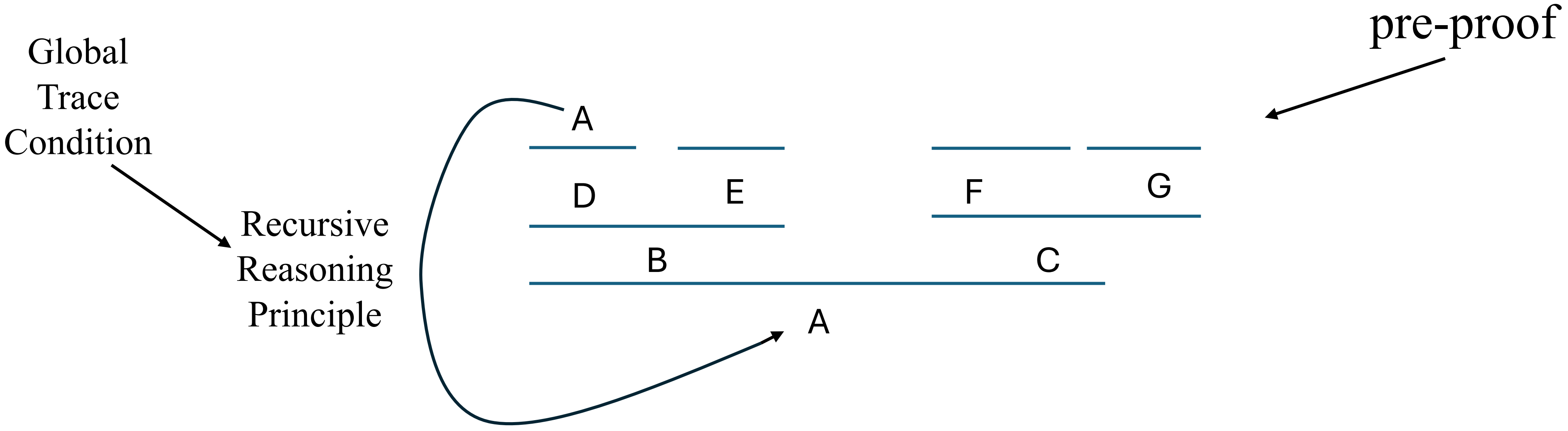
$$\cfrac{\cfrac{\cfrac{}{A \vdash A}\,\text{Atom}}{\vdash A, \neg A}\,\neg R}{\vdash A \vee \neg A}\,\vee R$$

Law of excluded middle

Cyclic Proof Systems

Cyclic Proof System

A cyclic proof system is a proof system that allows using recursive reasoning via back-links:



Cyclic Proof Systems

- Cyclist (Brotherston et. al., “A Generic Cyclic Theorem Prover”):
    - Generic Inductive Cyclic Proof System
- Das and Pous, “A Cut-Free Cyclic Proof System for Kleene Algebra”:
    - Cyclic Proof System for Kleene Algebra
- Afshari and Wehr, “Abstract Cyclic Proofs”:
    - Cyclic Proof System for Modal μ-Calculus via non-wellfounded proof theory

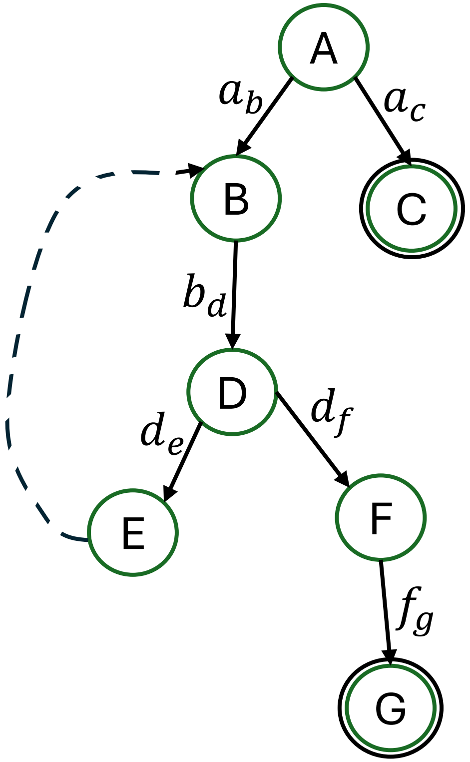
Goal Oriented Proof Search

- Proof Constructed from the bottom up
  - Begin at the goal and work backwards
- Iteratively expand the incomplete proof one leaf at a time
  - Pick some leaf that isn't an axiom or have a backlink
  - Try to match leaf with ancestor
    - Find ancestor with same sequent
    - Ensure global trace condition is preserved
      - E.g., by finding appropriate invairants and/or proving well-foundedness
  - Apply sequent rule

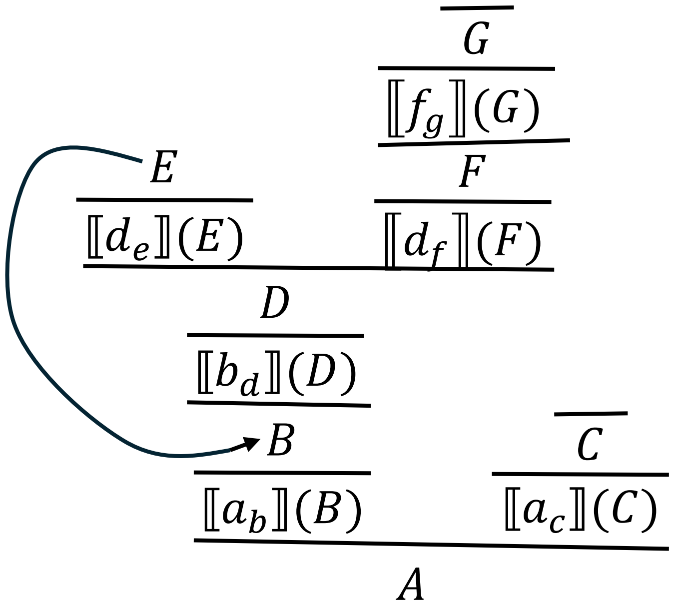
[Tsukada and Unno. "Software Model-Checking as Cyclic-Proof Search." POPL 2022.]

Other Techniques as Cyclic Proof Search

Original View



Cyclic Proof View



Other Techniques as Cyclic Proof Search

- Property Directed Reachability
  - Satisfiability of Constrained Horn Clauses
  - Program Safety (via Impact algorithm) [McMillan, "Lazy abstraction with interpolants."]
- Strategy Synthesis Algorithms
  - Reachability Games [Kincaid and Farzan, "Strategy Synthesis for Linear Arithmetic Games."]
  - Simulation Games [Murphy and Kincaid, "Relational Verification via Simulation Synthesis."]
- Symbolic Execution and Bounded Model Checking