Alternation Is Strict For Higher-Order Modal Fixpoint Logic

Alternating Krivine Automata and Alternation

Florian Bruse

University of Kassel, Germany

September 15, 2016

Motivation

HFL: Higher-order Modal Fixpoint Logic: $\mathcal{L}_\mu$ + simply typed lambda calculus

Alternating Parity Krivine Automata (APKA): Operational semantics for HFL

Motivation:

- enables local model-checking techniques
- automaton-based characterization of alternation
- possible intermediate to higher-order recursion schemes

Types

simple types given via $\tau \quad ::= \quad \mathsf{Pr} \mid \tau \to \tau$

because of right-associativity: $\tau = \tau_1 \to \ldots \to \tau_m \to \mathsf{Pr}$

each type induces a complete lattice over transition system $\mathcal{T} = (\mathcal{S}, \to, L)$ using pointwise orderings $\sqsubseteq$

$$\begin{aligned} [\![\mathsf{Pr}]\!] &:= (2^{\mathcal{S}}, \subseteq) \\ [\![\sigma \to \tau]\!] &:= ([\![\sigma]\!] \to_{\mathsf{mon.}} [\![\tau]\!], \sqsubseteq) \end{aligned}$$

Syntax of HFL

HFL = modal μ-calculus + simply typed λ-calculus

[Viswanathan² '04]

$$\varphi \ ::= \ q \mid \neg q \mid X \mid \varphi \vee \varphi \mid \langle a \rangle \varphi \mid \mu(X : \tau).\varphi \mid \lambda(X : \tau).\varphi \mid \varphi \, \varphi$$

plus duals $\varphi \wedge \psi$, $[a]\varphi$, $\nu(X : \tau)$ *natively*

well-formedness condition given by type system (not given here)

NB: can allow negation on arbitrary formulae, cope with extended type system

An Example Formula

Consider $(\mu X.\lambda x.\, x \vee (X\,[a]x))\,P$.

Unfolding via $\sigma X.\psi = \psi[\sigma X.\psi/X]$ yields
$\Big(\lambda x.\, x \vee (\mu X.\lambda x'.\, x' \vee (X\,[a]x'))\,[a]x\Big)\,P$.

Using β-reduction we get
$P \vee \big(\mu X.\lambda x'.\, x' \vee (X\,[a]x')\big)\,[a]P$.

More unfolding:
$P \vee (\lambda x'.\, x' \vee \big(\mu X.\lambda x''.\, x'' \vee (X\,[a]x'')\big)\,[a]x')\,[a]P$.

More β-reduction:
$P \vee [a]P \vee \big(\mu X.\lambda x''.\, x'' \vee (X\,[a]x'')\big)\,[a][a]P)$.

We get: $P \vee [a]P \vee [a][a]P \vee \ldots = \bigvee_{i=0}^{n}[a]^i P$

uniform inevitability!

Operational Semantics for HFL

proposed automaton model: Alternating Parity Krivine Automata (APKA)

- alternation for Boolean and modal operators ($\vee, \wedge, \langle a \rangle, [b]$)
- (stair-)parity condition for fixpoints
- Krivine Abstract Machine for higher-order features

challenge: get acceptance condition right, i.e., synchronize parity condition with Krivine machine

Alternating Parity Krivine Automata

APKA of index m is $\mathcal{A} = (\mathcal{X}, \delta, I, \Lambda, (\tau_X)_{X \in \mathcal{X}})$ where

- finite set of (fixpoint) states $\mathcal{X} = \{X_1, \dots, X_n\}$
- priority function $\Lambda \colon \mathcal{X} \to [1, m]$, resp. $[0, m-1]$
- transition function $\delta \colon X \mapsto \varphi_X$, generated from

$$\psi ::= P \mid \neg P \mid \psi \wedge \psi \mid \psi \vee \psi \mid \langle a \rangle \psi \mid [a]\psi \mid f_i^X \mid X' \mid (\psi\,\psi)$$

  where f_i^X of type τ_i^X for $i \leq n_X$ and φ_X of type τ_X.
- assignment of argument and value types

$$\tau_X = \tau_1^X \to \cdots \to \tau_{n_X}^X \to \mathrm{Pr}$$

- $I \in \mathcal{X}$ initial state with $\tau_I = \mathrm{Pr}$

state space is $\mathcal{Q} = \mathcal{X} \cup \bigcup_{X \in \mathcal{X}} \mathrm{sub}(\delta(X))$

Environments and Closures

environments handle variable lookup

$$e ::= e_0 \mid e = (f_1^X \mapsto (\psi_1, e_1), \ldots, f_{n_X}^X \mapsto (\psi_{n_X}, e_{n_X}), e')$$

e' is parent environment

(ψ, e_i) called closure

variable lookup:

$$e(f) = \begin{cases} (\psi_i, e_i) & \text{, if } f = f_i^X \\ \text{undefined} & \text{, otherwise} \end{cases}$$

Configurations

APKA accept LTS; explained in terms of 2-player game on configurations of the form

$$C \;=\; (s, (\psi, e), e', \Gamma, \Delta)$$

where

- s is current state in LTS
- (ψ, e) current closure with $\psi \in \mathcal{Q}$, e environment
- e' distinguished environment (point of current computation)
- $\Gamma = (\psi_n, e_n), \ldots, (\psi_1, e_1)$ stack of closures
- Δ stack of priorities

only use well-formed configurations (all environments defined etc.)

Acceptance of APKA

run over $\mathcal{T}, s_0$ starts in $(s_0, (I, e_0), e_0, \epsilon, \epsilon)$

game played between **V** and **R**:

- players move as per the transition relation (below)
- automaton accepts structure if **V** wins
- player who gets stuck loses
- infinite plays $\rightsquigarrow$ stair-parity condition on sequence of priority stacks

transition from $(s, (\psi, e), e', \Gamma, \Delta)$ depending on ψ

- $\psi = P$ or $\psi = \neg P$: **V** wins iff $\mathcal{T}, s \models \psi$
- $\psi = \psi_1 \vee \psi_2$: **V** chooses i, continue at $(s, (\psi_i, e), e', \Gamma, \Delta)$
- $\psi = [a]\psi'$: **R** chooses $s \xrightarrow{a} t$, cont. at $(t, (\psi', e), e', \Gamma, \Delta)$
- ...

More Game Moves

transition from $(s, (\psi, e), e', \Gamma, \Delta)$ depending on ψ

- $\psi = (\psi_1\, \psi_2)$: continue at $(s, (\psi_1, e), e', \Gamma \cdot (\psi_2, e), \Delta)$
- $\psi = X$: continue $(s, (\delta(X), e''), e'', \epsilon, \Delta \cdot \Lambda(X))$ where $\Gamma = C_1, \ldots, C_{n_X}$ and $e'' = (f_1^X \mapsto C_1, \ldots, f_{n_X}^X \mapsto C_{n_X}, e')$ new
- $\psi = f$ not of ground type: go to $(s, e(f), e', \Gamma, \Delta)$
- $\psi = f$ of ground type : go to $(s, (\psi', e''), e'', \Gamma, \Delta')$ where $e(f) = (\psi', e'')$ and Δ' is Δ with as many priorities removed as are between e' and e''

special role for ground type variables: undo priorities until "caller" is reached

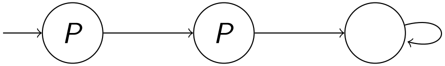
An Example

Consider $\mathcal{A} = (\mathcal{X}, \Lambda, I, \delta, (\tau_X)_{X \in \mathcal{X}})$ with

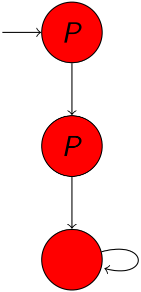
- $\mathcal{X} = \{I, X, Y\}$
- $\tau_I = \tau_Y = \mathsf{Pr}$, $\tau_X = \mathsf{Pr} \to \mathsf{Pr}$
- $\Lambda(I) = \Lambda(X) = 1, \Lambda(Y) = 0$
- $\delta(I) = \emptyset \mapsto (X\, \neg P)$
- $\delta(X) = x \colon \mathsf{Pr} \mapsto (\Diamond x) \vee \Box Y$
- $\delta(Y) = \emptyset \mapsto (X\ Y)$

Equivalent to $\Big(\mu X.\lambda x.\, \Diamond x \vee \Box \nu Y.(X\ Y)\Big)\, \neg P$

Run over (tree-unfolding of) this structure:



An Example Run



$$
\begin{aligned}
C_0 &= (s_1, (I, e_0), e_0, \epsilon, \epsilon) \\
C_1 &= (s_1, ((X\ \neg P), e_0), e_0, \epsilon, 1) \\
C_2 &= (s_1, (X, e_0), e_0, (\neg P, e_0), 1) \\
C_3 &= (s_1, (((\Diamond x) \vee \Box Y), e_1), e_1, \epsilon, 11) \\
C_4 &= (s_1, ((\Box Y), e_1), e_1, \epsilon, 11) \\
C_5 &= (s_2, (Y, e_1), e_1, \epsilon, 11) \\
C_6 &= (s_2, ((X\ Y), e_2), e_2, \epsilon, 110) \\
C_7 &= (s_2, (X, e_2), e_2, (Y, e_2), 110) \\
C_8 &= (s_2, (((\Diamond x) \vee \Box Y), e_3), e_3, \epsilon, 1101) \\
C_9 &= (s_2, ((\Diamond x), e_3), e_3, \epsilon, 1101) \\
C_{10} &= (s_3, (x, e_3), e_3, \epsilon, 1101) \\
C_{11} &= (s_3, (Y, e_2), e_2, \epsilon, 110)
\end{aligned}
$$

$$
\begin{aligned}
e_1 &= (x \mapsto (\neg P, e_0), e_0) \\
e_2 &= (\epsilon, e_1) \\
e_3 &= (x \mapsto (Y, e_2), e_2)
\end{aligned}
$$

Fixpoint Alternation

higher-order does not conquer fixpoint alternation

Theorem 1

For every $m \geq 2$ there is an APKA $\mathcal{A}_m$ index m that is not equivalent to any APKA of index $< m$.

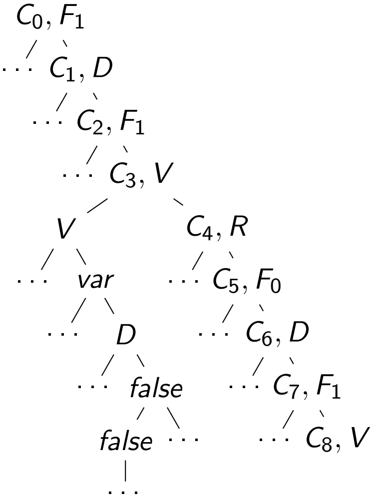
NB: $\mathcal{A}_m$ independent of type order

also induces alternation hierarchy on HFL

Sketch of the proof

- F.a. m fix suitable vocabulary τ_m and restrict to fully binary infinite trees
- Take game tree for acceptance game of a run of order-m automaton as underlying set of new LTS
- Annotate (via propositions) nodes in tree (configurations in game) depending on who moves, parity stack operations $\rightsquigarrow$ new tree $\mathcal{T}$.

Game Tree



$$C_0 = (s_1, (I, e_0), e_0, \epsilon, \epsilon)$$

$$C_1 = (s_1, ((X\ \neg P), e_0), e_0, \epsilon, 1)$$

$$C_2 = (s_1, (X, e_0), e_0, (\neg P, e_0), 1)$$

$$C_3 = (s_1, (((\Diamond x) \vee \Box Y), e_1), e_1, \epsilon, 11)$$

$$C_4 = (s_1, ((\Box Y), e_1), e_1, \epsilon, 11)$$

$$C_5 = (s_2, (Y, e_1), e_1, \epsilon, 11)$$

$$C_6 = (s_2, ((X\ Y), e_2), e_2, \epsilon, 110)$$

$$C_7 = (s_2, (X, e_2), e_2, (Y, e_2), 110)$$

$$C_8 = (s_2, (((\Diamond x) \vee \Box Y), e_3), e_3, \epsilon, 1101)$$

Sketch of the proof

- F.a. m fix suitable vocabulary τ_m and restrict to fully binary infinite trees
- Take game tree of acceptance game of a run of order-m automaton as underlying set of new LTS
- Annotate (via propositions) nodes in tree (configurations in game) depending on who moves, parity stack operations $\rightsquigarrow$ new tree $\mathcal{T}$.
- F.a. m there is fixed $\mathcal{A}_m$ s.t. $\mathcal{T} \models \mathcal{A}_m$ iff **V** wins underlying game
- This operation is contraction on metric space of f.b.i. trees $\rightsquigarrow$ obtain fixpoint via Banach Fixpoint Theorem
- No automaton of index $< m$ can be equivalent to $\mathcal{A}_m$ over this fixpoint