Alternation Is Strict For Higher-Order Modal Fixpoint Logic

Alternating Krivine Automata and Alternation

Florian Bruse

University of Kassel, Germany

September 15, 2016
Florian Bruse: APKA

**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 \( \tau ::= \text{Pr} \mid \tau \to \tau \)

because of right-associativity: \( \tau = \tau_1 \to \ldots \to \tau_m \to \text{Pr} \)

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

\[
\begin{align*}
[\text{Pr}] & := (2^S, \subseteq) \\
[\sigma \to \tau] & := ([\sigma] \to_{\text{mon.}} [\tau], \sqsubseteq)
\end{align*}
\]
Syntax of HFL

HFL = modal $\mu$-calculus + simply typed $\lambda$-calculus

[Viswanathan\textsuperscript{2} '04]

$$\varphi ::= q | \neg q | X | \varphi \lor \varphi | \langle a \rangle \varphi | \mu(X : \tau).\varphi | \lambda(X : \tau).\varphi | \varphi \varphi$$

plus duals $\varphi \land \psi$, $[a]\varphi$, $\nu(X : \tau)$ 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
\[
\left(\lambda x. x \vee (\mu X. \lambda x'. x' \vee (X \;[a]x')) \;[a]x\right) \;P.
\]

Using \(\beta\)-reduction we get
\[
P \vee (\mu X. \lambda x'. x' \vee (X \;[a]x')) \;[a]P.
\]

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

More \(\beta\)-reduction:
\[
P \vee [a]P \vee (\mu X. \lambda x'''. x''' \vee (X \;[a]x''')) \;[a][a]P).
\]

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 ($\lor, \land, \langle a \rangle, \lbrack b \rbrack$)
- (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 $\mathcal{X} = \{X_1, \ldots, X_n\}$
- priority function $\Lambda : \mathcal{X} \rightarrow [1, m]$, resp. $[0, m - 1]$.
- transition function $\delta : X \mapsto \varphi_X$, generated from

$$\psi ::= P \mid \neg P \mid \psi \land \psi \mid \psi \lor \psi \mid \langle a \rangle \psi \mid [a] \psi \mid f_i^X \mid X' \mid (\psi \psi)$$

where $f_i^X$ of type $\tau_i^X$ for $i \leq n_X$ and $\varphi_X$ of type $\tau_X$.

- assignment of argument and value types

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

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

state space is $Q = \mathcal{X} \cup \bigcup_{X \in \mathcal{X}} \text{sub}(\delta(X))$
**Environments and Closures**

Environments handle variable lookup

\[ e ::= e_0 \mid e = (f_1^X \mapsto (\psi_1, e_1), \ldots, f_n^X \mapsto (\psi_n, e_n), e') \]

\( e' \) is parent environment

\((\psi, e_i)\) called closure

Variable lookup:

\[ e(f) = \begin{cases} (\psi_i, e_i) & \text{, if } f = f_i^X \\ \text{undefined} & \text{, otherwise} \end{cases} \]
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
- \((\psi, e)\) current closure with \(\psi \in Q\), \(e\) environment
- \(e'\) distinguished environment (point of current computation)
- \(\Gamma = (\psi_n, e_n), \ldots, (\psi_1, e_1)\) stack of closures
- \(\Delta\) stack of priorities

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

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

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 $\Rightarrow$ stair-parity condition on sequence of priority stacks

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

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

- \(\psi = \left(\psi_1 \psi_2\right)\): 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 \(e'' = (f_{X}^1 \mapsto C_1, \ldots, f_{X}^{n_X} \mapsto C_{n_X}, e')\) 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 \(\Delta'\) is \(\Delta\) 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

- $\mathcal{X} = \{I, X, Y\}$
- $\tau_I = \tau_Y = \text{Pr}, \tau_X = \text{Pr} \rightarrow \text{Pr}$
- $\Lambda(I) = \Lambda(X) = 1, \Lambda(Y) = 0$
- $\delta(I) = \emptyset \mapsto (X \neg P)$
- $\delta(X) = x: \text{Pr} \mapsto (\lozenge x) \lor \square Y$
- $\delta(Y) = \emptyset \mapsto (X Y)$

Equivalent to $\left( \mu X. \lambda x. \lozenge x \lor \square \nu Y. (X Y) \right) \neg P$

Run over (tree-unfolding of) this structure:
An Example Run

\[ C_0 = (s_1, (I, e_0), e_0, \varepsilon, \varepsilon) \]
\[ C_1 = (s_1, ((X \neg P), e_0), e_0, \varepsilon, 1) \]
\[ C_2 = (s_1, (X, e_0), e_0, (\neg P, e_0), 1) \]
\[ C_3 = (s_1, (((\Diamond x) \lor \Box Y), e_1), e_1, \varepsilon, 11) \]
\[ C_4 = (s_1, ((\Box Y), e_1), e_1, \varepsilon, 11) \]
\[ C_5 = (s_2, (Y, e_1), e_1, \varepsilon, 11) \]
\[ C_6 = (s_2, ((X Y), e_2), e_2, \varepsilon, 110) \]
\[ C_7 = (s_2, (X, e_2), e_2, (Y, e_2), 110) \]
\[ C_8 = (s_2, (((\Diamond x) \lor \Box Y), e_3), e_3, \varepsilon, 1101) \]
\[ C_9 = (s_2, ((\Diamond x), e_3), e_3, \varepsilon, 1101) \]
\[ C_{10} = (s_3, (x, e_3), e_3, \varepsilon, 1101) \]
\[ C_{11} = (s_3, (Y, e_2), e_2, \varepsilon, 110) \]
Fixpoint Alternation

higher-order does not conquer fixpoint alternation

**Theorem 1**

For every $m \geq 2$ there is an APKA $A_m$ index $m$ that is not equivalent to any APKA of index $< m$.

NB: $A_m$ independent of type order

also induces alternation hierarchy on HFL
Sketch of the proof

- F.a. $m$ fix suitable vocabulary $\tau_m$ 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 $\leadsto$ new tree $T$.
Game Tree

\[ C_0, F_1 \]
\[ \quad / \quad \backslash \]
\[ \quad \ldots \quad C_1, D \]
\[ \quad / \quad \backslash \]
\[ \quad \ldots \quad C_2, F_1 \]
\[ \quad / \quad \backslash \]
\[ \quad \ldots \quad C_3, V \]
\[ \quad / \quad \backslash \quad / \quad \backslash \]
\[ \quad V \quad C_4, R \]
\[ \quad / \quad \backslash \quad / \quad \backslash \]
\[ \quad \ldots \quad \text{var} \quad \ldots \quad C_5, F_0 \]
\[ \quad / \quad \backslash \quad / \quad \backslash \]
\[ \quad \ldots \quad D \quad \ldots \quad C_6, D \]
\[ \quad / \quad \backslash \quad / \quad \backslash \]
\[ \quad \ldots \quad \text{false} \quad \ldots \quad C_7, F_1 \]
\[ \quad / \quad \backslash \quad / \quad \backslash \]
\[ \quad \text{false} \quad \ldots \quad \ldots \quad C_8, V \]
\[ \quad | \]
\[ \quad \ldots \]

\[ C_0 = (s_1, (I, e_0), e_0, \epsilon, \epsilon) \]
\[ C_1 = (s_1, ((X \neg P), e_0), e_0, \epsilon, 1) \]
\[ C_2 = (s_1, (X, e_0), e_0, (\neg P, e_0), 1) \]
\[ C_3 = (s_1, (((\lozenge x) \lor \Box Y), e_1), e_1, \epsilon, 11) \]
\[ C_4 = (s_1, ((\Box Y), e_1), e_1, \epsilon, 11) \]
\[ C_5 = (s_2, (Y, e_1), e_1, \epsilon, 11) \]
\[ C_6 = (s_2, ((X Y), e_2), e_2, \epsilon, 110) \]
\[ C_7 = (s_2, (X, e_2), e_2, (Y, e_2), 110) \]
\[ C_8 = (s_2, (((\lozenge x) \lor \Box Y), e_3), e_3, \epsilon, 1101) \]
Sketch of the proof

- F.a. $m$ fix suitable vocabulary $\tau_m$ 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 $T$.
- F.a. $m$ there is fixed $A_m$ s.t. $T \models A_m$ 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 $A_m$ over this fixpoint