Specifying Program Properties Using Modal Fixpoint Logics

Martin Lange

University of Kassel, Germany

8th Indian Conference on Logic and its Applications
04/03/19
1 Motivation

2 Specifying Properties using Modal Fixpoint Logic
   - The Modal $\mu$-Calculus
   - Higher-Order Fixpoint Logic
   - Computational Complexity and Decidability
   - Automata, Logic, Games
   - Fixpoint Quantifier Alternation
   - Polyadic Higher-Order Fixpoint Logic

3 Future Work / Open Questions
Verification of Reactive Systems

general motivation: formal verification of dynamic systems

typical ICT systems are **reactive**: 
Requirements for Specification Languages

generally needed for \textit{formal} verification: \textit{formal specification} languages, i.e. logics

especially needed for specifying properties of reactive systems: to speak about . . .

\begin{itemize}
  \item . . . immediate behaviour: modal operators
    \textit{“it is possible to react to any input of the form . . .”}
    \[\leadsto \diamondsuit \varphi, \square \varphi\]
  \item . . . behaviour in the infinite: limit operators
    \textit{“every request is eventually granted”}
    convenient tool: least and greatest \textit{fixpoints}
    \[\leadsto \mu X. \varphi(X), \nu X. \varphi(X)\]
\end{itemize}
The Modal $\mu$-Calculus

multi-modal logic + extremal fixpoint quantifiers

$$\varphi ::= p \mid X \mid \varphi \lor \varphi \mid \neg \varphi \mid \langle a \rangle \varphi \mid \mu X.\varphi$$

usual abbreviations: $\varphi \land \psi$, $\varphi \rightarrow \psi$, $[a] \varphi := \neg \langle a \rangle \neg \varphi$, $\nu X.\varphi ::= \neg \mu X.\neg \varphi[\neg X/X]$
The Modal $\mu$-Calculus

multi-modal logic + extremal fixpoint quantifiers

$$\phi ::= p \mid X \mid \phi \lor \phi \mid \neg \phi \mid \langle a \rangle \phi \mid \mu X . \phi$$

usual abbreviations: $\phi \land \psi$, $\phi \rightarrow \psi$, $[a] \phi := \neg \langle a \rangle \neg \phi$, $\nu X . \phi := \neg \mu X . \neg \phi [\neg X / X]$

interpreted over transition system

$T = (S, \{\overset{a}{\rightarrow} \mid a \in A\}, L : S \rightarrow 2^P)$

semantics usually given as $[\phi]_T^\rho \subseteq S$ with Knaster-Tarski
Examples

typical $\mathcal{L}_\mu$-definable properties:

- $\nu X.\langle a \rangle X$
Examples

typical $L_\mu$-definable properties:

- $\nu X. \langle a \rangle X$

- $\mu X. p \lor (\diamond t t \land \Box X) \ (\equiv \ AFp \ \text{in CTL})$
Examples

Typical $\mathcal{L}_\mu$-definable properties:

- $\nu X. \langle a \rangle X$
- $\mu X. p \lor (\diamond t t \land \Box X)$ ($\equiv A F p$ in CTL)
- $\mu X. [a] X$
Examples

typical $\mathcal{L}_\mu$-definable properties:

- $\nu X. \langle a \rangle X$
- $\mu X. p \lor (\lozenge \top \land \Box X) \equiv AFp$ in CTL
- $\mu X. [a] X$
- $\nu X. \mu Y. \lozenge ((p \land X) \lor Y)$
The Expressive Power of $\mathcal{L}_\mu$

**Theorem 1** *(Emerson/Jutla ’88; Janin/Walukiewicz ’96)*

A bisimulation-invariant tree language is $\mathcal{L}_\mu$-definable iff it is regular
The Expressive Power of $\mathcal{L}_\mu$

**Theorem 1** (Emerson/Jutla ’88; Janin/Walukiewicz ’96)

A bisimulation-invariant tree language is $\mathcal{L}_\mu$-definable iff it is regular

... typical properties that are not $\mathcal{L}_\mu$-definable:

- uniform inevitability,
  something holds on all paths at the same time
The Expressive Power of $\mathcal{L}_\mu$

**Theorem 1 (Emerson/Jutla ’88; Janin/Walukiewicz ’96)**

A bisimulation-invariant tree language is $\mathcal{L}_\mu$-definable iff it is regular

typical properties that are not $\mathcal{L}_\mu$-definable:

- uniform inevitability, something holds on all paths at the same time
- unlimited counting like IO-buffer properties
The Expressive Power of $L_\mu$

**Theorem 1 (Emerson/Jutla ’88; Janin/Walukiewicz ’96)**

A bisimulation-invariant tree language is $L_\mu$-definable iff it is regular.

typical properties that are not $L_\mu$-definable:

- **uniform inevitability**, something holds on all paths at the same time
- unlimited *counting* like IO-buffer properties
- repetitions of unbounded sequences of actions
The Expressive Power of $\mathcal{L}_\mu$

**Theorem 1** (Emerson/Jutla ’88; Janin/Walukiewicz ’96)

A bisimulation-invariant tree language is $\mathcal{L}_\mu$-definable iff it is regular.

Typical properties that are not $\mathcal{L}_\mu$-definable:

- uniform inevitability, something holds on all paths at the same time
- unlimited counting like IO-buffer properties
- repetitions of unbounded sequences of actions
- ...
1 Motivation

2 Specifying Properties using Modal Fixpoint Logic
   - The Modal $\mu$-Calculus
   - Higher-Order Fixpoint Logic
   - Computational Complexity and Decidability
   - Automata, Logic, Games
   - Fixpoint Quantifier Alternation
   - Polyadic Higher-Order Fixpoint Logic

3 Future Work / Open Questions
Types

we need a simple type system with variances

$$\tau ::= \text{Pr} \mid \tau^v \rightarrow \tau$$

$$\nu ::= + \mid - \mid 0$$

because of right-associativity: $$\tau = \tau^{v_1}_1 \rightarrow \ldots \rightarrow \tau^{v_m}_m \rightarrow \text{Pr}$$
Types

we need a simple type system with variances

\[\tau ::= \text{Pr} \mid \tau^\nu \to \tau\]
\[\nu ::= + \mid - \mid 0\]

because of right-associativity: \(\tau = \tau_1^{\nu_1} \to \ldots \to \tau_m^{\nu_m} \to \text{Pr}\)

for a partial order \(V = (M, \sqsubseteq)\) let

\[V^+ := (M, \sqsubseteq) \quad V^- := (M, \sqsupseteq) \quad V^0 := (M, =)\]
Types

we need a simple type system with variances

\[ \tau ::= \Pr | \tau^v \to \tau \]

\[ v ::= + | - | 0 \]

because of right-associativity: \( \tau = \tau_1^{v_1} \to \ldots \to \tau_m^{v_m} \to \Pr \)

for a partial order \( V = (M, \sqsubseteq) \) let

\[ V^+ := (M, \sqsubseteq) \quad V^- := (M, \supseteq) \quad V^0 := (M, =) \]

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

\[
\begin{align*}
[[\Pr]] & := (2^S, \subseteq) \\
[[\sigma^v \to \tau]] & := ([[\sigma]]^\rightarrow_{\text{monotone}} [[\tau]], \sqsubseteq)
\end{align*}
\]
Formulas

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

[Viswanathan$^2$ ’04]

$$\varphi ::= p \mid X \mid \varphi \lor \varphi \mid \neg \varphi \mid \langle a \rangle \varphi \mid \mu X .\varphi \mid \lambda X .\varphi \mid \varphi \varphi$$
Formulas

\[ \text{HFL} = \text{modal } \mu\text{-calculus} + \text{simply typed } \lambda\text{-calculus} \]

\[ \phi ::= p \mid X \mid \phi \lor \phi \mid \neg \phi \mid \langle a \rangle \phi \mid \mu(X : \tau).\phi \mid \lambda(X^v : \tau).\phi \mid \phi \phi \]

well-formedness condition given by type system

needed to exclude \( \langle a \rangle q \langle b \rangle p, \mu X.\neg X \), etc.
**Formulas**

\[ \text{HFL} = \text{modal } \mu\text{-calculus } + \text{ simply typed } \lambda\text{-calculus} \]

[Viswanathan\textsuperscript{2} ’04]

\[ \varphi ::= p \mid X \mid \varphi \lor \varphi \mid \neg \varphi \mid \langle a \rangle \varphi \mid \mu(X : \tau).\varphi \mid \lambda(X^\nu : \tau).\varphi \mid \varphi \varphi \]

well-formedness condition given by type system needed to exclude \( \langle a \rangle q \langle b \rangle p, \mu X.\neg X \), etc.

often use more convenient syntax, e.g.

\[ \mu F(X, g).\neg X \lor F(g(X), g^2) \]

instead of

\[ \mu(F : \text{Pr}^- \to (\text{Pr}^+ \to \text{Pr})^+ \to \text{Pr}).\lambda(X : \text{Pr}).\lambda(g : \text{Pr}^+ \to \text{Pr}).\neg X \lor F(gX)(\lambda(Y : \text{Pr}^+).g(gY))) }\]
Negation is Trickier

why not simple condition as in the modal $\mu$-calculus

\textit{every fixpoint variable occurs under an even number of negation symbols in its defining fixpoint formula}

e.g. $\neg \mu X. \neg \mu Y. \langle a \rangle \neg X \lor \langle b \rangle Y$
Negation is Trickier

why not simple condition as in the modal $\mu$-calculus

*every fixpoint variable occurs under an even number of negation symbols in its defining fixpoint formula*

e.g. $\neg\mu X.\neg\mu Y.(a)\neg X \lor (b) Y$

$\lambda$-abstraction can shift negations into different branches of the syntax tree, e.g. $\mu X.(\lambda Y.\neg Y) X$

this formula is not well-formed
The Typing Rules

\[ \varphi \] well-formed iff \( \emptyset \vdash \varphi : \Pr \) is derivable

\[
\begin{align*}
\Gamma &\vdash \varphi : \Pr & \nu \in \{0, +\} &\quad \Gamma, X^\nu : \tau \vdash X : \tau \\
\Gamma &\vdash \neg \varphi : \Pr & \Gamma &\vdash \varphi : \Pr \\
\Gamma, X^\nu : \sigma &\vdash \varphi : \tau & \Gamma &\vdash \lambda (X^\nu : \sigma). \varphi : (\sigma^\nu \rightarrow \tau) \\
\Gamma &\vdash \varphi : (\sigma^+ \rightarrow \tau) & \Gamma &\vdash \varphi \psi : \tau \\
\Gamma &\vdash \varphi : (\sigma^- \rightarrow \tau) & \Gamma &\vdash \neg \varphi : \Pr \\
\Gamma &\vdash \varphi : (\sigma^0 \rightarrow \tau) & \Gamma &\vdash \varphi \psi : \tau \\
\Gamma, X^+ : \tau &\vdash \varphi : \tau & \Gamma &\vdash \mu (X : \tau). \varphi : \tau
\end{align*}
\]
Semantics of HFL

Semantics of formula $\varphi$ with $\emptyset \vdash \varphi : \tau$ is element of $[[\tau]]$ over transition system $\mathcal{T} = (S, \rightarrow, L)$

\[
\begin{align*}
[[\Gamma \vdash p : \text{Pr}]_\eta^\mathcal{T}] & = \{s \in S \mid p \in L(s)\} \\
[[\Gamma \vdash X : \tau]_\eta^\mathcal{T}] & = \eta(X) \\
[[\Gamma \vdash \neg\varphi : \text{Pr}]_\eta^\mathcal{T}] & = S \setminus [[\neg \vdash \varphi : \text{Pr}]_\eta^\mathcal{T}] \\
[[\Gamma \vdash \varphi : \sigma^\vee \rightarrow \tau]_\eta^\mathcal{T}] & = f \in [[\sigma^\vee \rightarrow \tau]] \text{ s.t. } \bar{f} = [[\neg \vdash \varphi : \sigma^\vee \rightarrow \tau]]_\eta^\mathcal{T} \\
[[\Gamma \vdash \varphi \vee \psi : \text{Pr}]_\eta^\mathcal{T}] & = [[\Gamma \vdash \varphi : \text{Pr}]_\eta^\mathcal{T}] \cup [[\Gamma \vdash \psi : \text{Pr}]_\eta^\mathcal{T}] \\
[[\Gamma \vdash \langle a \rangle \varphi : \text{Pr}]_\eta^\mathcal{T}] & = \{s \in S \mid s \xrightarrow{a} t \text{ for some } t \in [[\Gamma \vdash \varphi : \text{Pr}]]_\eta^\mathcal{T}\} \\
[[\Gamma \vdash \lambda(X^\vee : \sigma).\varphi : \sigma^\vee \rightarrow \tau]_\eta^\mathcal{T}] & = f \in [[\sigma^\vee \rightarrow \tau]] \text{ s.t. } \forall x \in [[\sigma]] \\
& \quad f \cdot x = [[\Gamma, X^\vee : \sigma \vdash \varphi : \tau]]_\eta^\mathcal{T}[X \mapsto x] \\
[[\Gamma \vdash \varphi \psi : \tau]_\eta^\mathcal{T}] & = [[\Gamma \vdash \varphi : \sigma^\vee \rightarrow \tau]]_\eta^\mathcal{T} \cup [[\Gamma \vdash \psi : \sigma]]_\eta^\mathcal{T} \\
[[\Gamma \vdash \mu(X : \tau)\varphi : \tau]_\eta^\mathcal{T}] & = \bigcap\{x \in [[\tau]] \mid [[\Gamma, X^+ : \tau \vdash \varphi : \tau]]_\eta^\mathcal{T}[X \mapsto x] \subseteq_\tau x\}
\end{align*}
\]
Semantics of HFL

semantics of formula $\varphi$ with $\emptyset \vdash \varphi : \tau$ is element of $[[\tau]]$ over transition system $T = (S, \rightarrow, L)$

$$[[\Gamma \vdash p : \text{Pr}]]^T_\eta = \{s \in S \mid p \in L(s)\}$$
$$[[\Gamma \vdash X : \tau]]^T_\eta = \eta(X)$$
$$[[\Gamma \vdash \neg \varphi : \text{Pr}]]^T_\eta = S \setminus [[\Gamma \vdash \varphi : \text{Pr}]]^T_\eta$$
$$[[\Gamma \vdash \varphi : \sigma^v \rightarrow \tau]]^T_\eta = f \in [\sigma^v \rightarrow \tau]$ s.t. $\vec{f} = [[\Gamma \vdash \varphi : \sigma^v \rightarrow \tau]]^T_\eta$.
$$[[\Gamma \vdash \varphi \lor \psi : \text{Pr}]]^T_\eta = [[\Gamma \vdash \varphi : \text{Pr}]]^T_\eta \cup [[\Gamma \vdash \psi : \text{Pr}]]^T_\eta$$
$$[[\Gamma \vdash \langle a \rangle \varphi : \text{Pr}]]^T_\eta = \{s \in S \mid s \xrightarrow{a} t \text{ for some } t \in [[\Gamma \vdash \varphi : \text{Pr}]]^T_\eta\}$$
$$[[\Gamma \vdash \lambda(X^v : \sigma).\varphi : \sigma^v \rightarrow \tau]]^T_\eta = f \in [\sigma^v \rightarrow \tau]$ s.t. $\forall x \in [\sigma]$
$$f \ x = [[\Gamma, X^v : \sigma \vdash \varphi : \tau]]^T_\eta[X \mapsto x]$$
$$[[\Gamma \vdash \varphi \psi : \tau]]^T_\eta = [[\Gamma \vdash \varphi : \sigma^v \rightarrow \tau]]^T_\eta \ [\Gamma' \vdash \psi : \sigma]^T_\eta$$
$$[[\Gamma \vdash \mu(X : \tau)\varphi : \tau]]^T_\eta = \{x \in [\tau] \mid [[\Gamma, X^+: \tau \vdash \varphi : \tau]]^T_\eta[X \mapsto x] \sqsubseteq \tau \ x\}$$

Prop. 1: $(\lambda(X : \tau).\varphi) \psi \equiv \varphi[\psi/X]$ (β-reduction)

Prop. 2: $\mu(X : \tau).\varphi \equiv \varphi[(\mu(X : \tau).\varphi)/X]$ (fixpoint unfolding)
Examples

what properties are expressed by the following formulas?

$$(\mu F(X).X \lor \langle a \rangle F(\langle b \rangle X))$$
Examples

what properties are expressed by the following formulas?

\[(\mu F(X).X \lor F(\Box X)) \text{ ff}\]
Examples

what properties are expressed by the following formulas?

\[ \varphi_{\text{word}} := \neg \bigvee_{a \neq b} (\mu F(X, Y). (X \land Y) \lor F(\diamond X, \diamond Y)) \langle a \rangle \top \langle b \rangle \top \]
**Examples**

what properties are expressed by the following formulas?

$$(\mu F(g, g', g''). (g \circ g' \circ g'') \lor F(g \circ \langle a \rangle, g' \circ \langle b \rangle, g'' \circ \langle c \rangle)) \; id \; id \; id \; \top$$

where $id := \lambda X. X$, $\langle a \rangle := \lambda X. \langle a \rangle X$, and $f \circ g := \lambda X. f (g X)$
Examples

what properties are expressed by the following formulas?

\( (\nu F(X).[b]X \land [a]F(F(X))) \)
Examples

what properties are expressed by the following formulas?

\[
(\mu F(g). (g \circ g) \lor \bigvee_{a \in \Sigma} F(g \circ \langle a \rangle)) \text{id } \text{tt}
\]
Examples

what properties are expressed by the following formulas?

\[ \psi_m \psi_{m-1} \ldots \psi_1 \Box \Box \square \Box \text{ff} \text{ where } \psi_i := \lambda F. \lambda X. F (F X) \]
1 Motivation

2 Specifying Properties using Modal Fixpoint Logic
   - The Modal $\mu$-Calculus
   - Higher-Order Fixpoint Logic
   - Computational Complexity and Decidability
   - Automata, Logic, Games
   - Fixpoint Quantifier Alternation
   - Polyadic Higher-Order Fixpoint Logic

3 Future Work / Open Questions
Fragments by Type Order

type order: \( \text{ord}(\tau_1 \to \ldots \to \tau_m \to \text{Pr}) = \max\{1 + \text{ord}(\tau_i)\} \)

\( \text{HFL}^{k,m} = \) well-formed formulas using type annotations of order at most \( k \) and at most \( m \) arguments
Fragments by Type Order

type order: \( \text{ord}(\tau_1 \rightarrow \ldots \rightarrow \tau_m \rightarrow \Pr) = \max\{1 + \text{ord}(\tau_i)\} \)

\( \text{HFL}^{k,m} \) = well-formed formulas using type annotations of order at most \( k \) and at most \( m \) arguments

recall examples above:

order 1: “balanced tree”, “bisimilarity to a word”, all CFL path properties, some CSL path properties
Fragments by Type Order

type order: \( \text{ord}(\tau_1 \rightarrow \ldots \rightarrow \tau_m \rightarrow \text{Pr}) = \max\{1 + \text{ord}(\tau_i)\} \)

\( \text{HFL}^{k,m} \) = well-formed formulas using type annotations of order at most \( k \) and at most \( m \) arguments

recall examples above:

order 1: “balanced tree”, “bisimilarity to a word”, all CFL path properties, some CSL path properties

order 2: (all?) CSL path properties
Fragments by Type Order

type order: \(ord(\tau_1 \to \ldots \to \tau_m \to \text{Pr}) = \max\{1 + ord(\tau_i)\}\)

\(\text{HFL}^{k,m}\) = well-formed formulas using type annotations of order at most \(k\) and at most \(m\) arguments

recall examples above:

order 1: “balanced tree”, “bisimilarity to a word”, all CFL path properties, some CSL path properties

order 2: (all?) CSL path properties

order \(k\): measure path lengths up to \(2^2\ldots2^n\)
Theorem 2 (Axelsson/L./Somla '07)

For \( k \geq 1, m \geq 0 \): model checking \( \text{HFL}^{k,m} \) is \( k\text{-EXPTIME-compl.} \).

Proof sketch: (upper bounds) consider height of lattices \([\tau]\):

\[
\text{height}(\tau_1 \rightarrow \ldots \rightarrow \tau_m \rightarrow \text{Pr}) = (n + 1) \cdot \prod_{i=1}^{m} |[\tau_i]|
\]

with

\[
|[\tau_1 \rightarrow \ldots \tau_m \rightarrow \text{Pr}]| = 2^n \cdot \prod_{i=1}^{m} |[\tau_i]|
\]

\( \Rightarrow \) naïve bottom-up evaluation in time dominated by lattice height
Model Checking: Lower Bounds

for lower bounds: reduction from the word problem for alternating $(k-1)$-EXPSPACE Turing machines

main ingredients:

- representation of large numbers by (lexicographically ordered) functions
- stepwise counting in HFL

let $inc := \lambda X . X \leftrightarrow \Diamond \neg X$, what is $inc(\emptyset)$?
Model Checking: Lower Bounds

for lower bounds: reduction from the word problem for alternating $(k-1)$-EXPSPACE Turing machines

main ingredients:

- representation of large numbers by (lexicographically ordered) functions
- stepwise counting in HFL

let $inc := \lambda X. X \leftrightarrow \lozenge \neg X$, what is $inc(\emptyset)$, $inc^k(\emptyset)$ for $k > 1$?
Model Checking: Lower Bounds

for lower bounds: reduction from the word problem for alternating \((k-1)\)-EXPSPACE Turing machines

main ingredients:

- representation of large numbers by (lexicographically ordered) functions
- stepwise counting in HFL

let \( \text{inc} := \lambda X. X \leftrightarrow \Diamond \neg X \), what is \( \text{inc}(\emptyset) \), \( \text{inc}^k(\emptyset) \) for \( k > 1 \)?

principle extendable to higher orders using tests for equality, less-than, greater-than

\( \rightsquigarrow \) simulate run of space-bounded Turing machines
Tail Recursion

Def.: tail-recursive fragment trHFL intuitively: fixpoint variables of highest type . . .

- not in both conjuncts \[ \rightsquigarrow \text{no } \langle a \rangle X \langle b \rangle X \]
- not behind modal box operators \[ \rightsquigarrow \text{no } [a]X \]
- not in argument position \[ \rightsquigarrow \text{no } \lambda F.\lambda X.F(F(X)) \]

formal definition via type system [[Bruse ’18]]
Tail Recursion

**Def.:** tail-recursive fragment trHFL intuitively: fixpoint variables of highest type . . .

- not in both conjuncts \[\leadsto\] no \(\langle a\rangle X \langle b\rangle X\)
- not behind modal box operators \[\leadsto\] no \([a]X\)
- not in argument position \[\leadsto\] no \(\lambda F.\lambda X.F(F(X))\)

formal definition via type system [[Bruse ’18]]

---

**Theorem 3 (Bruse/L./Lozes ’17)**

*For \(k \geq 1, m \geq 0: \) model checking trHFL\(^k,m\) is \((k - 1)\)-EXPSPACE-complete*

**Proof:** lower bound: similar
upper bound: use nondeterministic top-down algorithm and Savitch’s Theorem
Undecidability of Satisfiability

Theorem 4

Satisfiability for $\text{HFL}^1$ is undecidable (at least $\Sigma_1^1$-hard)

follows from undecidability of Fixpoint Logic with Chop [Müller-Olm, ’99] and embedding into $\text{HFL}^1$ [Viswanathan², ’04]

undecidability not hard to see:

$$\varphi_{\text{word}} \land \bigvee_{w \in L(G_1)} \langle w \rangle_{tt} \land \bigvee_{w \in L(G_2)} \langle w \rangle_{tt}$$

expresses non-emptiness of intersection between CFGs $G_1$ and $G_2$
No Finite Model Property

decidability of model checking and $\Sigma^1_1$-hardness of satisfiability implies loss of finite model property

also possible to see directly

Theorem 5

$HFL^1$ does not have the finite model property.

Proof:

$\mu X. \Box X \land (\nu F(Y). Y \land F(\Diamond Y)) \land \top$

forbids infinite paths but requires paths of unbounded length

$\square$
1 Motivation

2 Specifying Properties using Modal Fixpoint Logic
   - The Modal $\mu$-Calculus
   - Higher-Order Fixpoint Logic
   - Computational Complexity and Decidability
   - Automata, Logic, Games
     - Fixpoint Quantifier Alternation
     - Polyadic Higher-Order Fixpoint Logic

3 Future Work / Open Questions
automata and games are important computational tools for temporal logics

**Theorem 6 (Stirling '95, Walukiewicz '96)**

*Model Checking μ-calculus = solving parity games.*

**Def.:** parity game is a 2-player game on graphs where nodes have priorities. Player **Verifier** wins infinite play iff outermost fixpoint seen infinitely often is of type ν

**Ex.:**

\[ \models (νX.μY.◊((p \land X) \lor Y) \]
Stair-Parity Games

HFL model checking is not a parity game

Ex.: consider

\[
\begin{array}{c}
\xrightarrow{a} \\
\end{array}
\quad \xrightarrow{b}
\]

\[
\models (\mu F(X). \langle b \rangle X \lor \langle a \rangle \nu G. F(G)) \quad \tt
\]
Stair-Parity Games

HFL model checking is not a parity game

Ex.: consider

\[
\begin{align*}
\text{a} & \quad \text{?} \\
\quad \downarrow & \quad \text{b}
\end{align*}
\]

\[\models (\mu F(X).\langle b\rangle X \lor \langle a\rangle \nu G.F(G)) \text{ tt}\]

refinement needed here

observation for HFL\(^{1,1}\): fixpoints have 1 argument \(\rightsquigarrow\) recursion call stack

Def.: stair-parity game is pushdown game with parity condition evaluated on persistent part of call stack

Theorem 7 (L. ’02, L. ’06)

Model checking HFL\(^{1,1}\) = stair-parity game
Games for HFL

for general HFL further extension needed; best formulated as abstract automaton model with acceptance game

proposed automaton model: Alternating Parity Krivine Automata (APKA)

- alternation for Boolean and modal operators ($\lor$, $\land$, $\langle a \rangle$, $[b]$)
for general HFL further extension needed; best formulated as abstract automaton model with acceptance game

proposed automaton model: Alternating Parity Krivine Automata (APKA)

- alternation for Boolean and modal operators ($\lor, \land, \langle a \rangle, [b]$)
- (stair-)parity condition for fixpoints
Games for HFL

for general HFL further extension needed; best formulated as abstract automaton model with acceptance game

proposed automaton model: Alternating Parity Krivine Automata (APKA)

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

for general HFL further extension needed; best formulated as abstract automaton model with acceptance game

proposed automaton model: Alternating Parity Krivine Automata (APKA)

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

challenge: get acceptance condition right, i.e. synchronise parity condition with Krivine machine
Alternating Parity Krivine Automata

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

- finite set of (fixpoint) states $\mathcal{X} = \{X_1, \ldots, X_n\}$
Alternating Parity Krivine Automata

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

- finite set of (fixpoint) states $X = \{X_1, \ldots, X_n\}$
- priority function $\Lambda : X \to [1, m]$, resp. $[0, m - 1]$
Alternating Parity Krivine Automata

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

- finite set of (fixpoint) states $\mathcal{X} = \{X_1, \ldots, X_n\}$
- priority function $\Lambda : \mathcal{X} \to [1, m]$, 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 $f_{i}^{X} : \tau_{i}^{X}$ for $i \leq n_X$ and $\varphi_X : \tau_X$.

state space is $Q = \mathcal{X} \cup \bigcup_{X \in \mathcal{X}} \text{sub}(\delta(X))$
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 $f_i^X : \tau_i^X$ for $i \leq n_X$ and $\varphi_X : \tau_X$.
- assignment of argument and value types

$$\tau_X = \tau_1^X \rightarrow \cdots \rightarrow \tau_{n_X}^X \rightarrow \tau_{n_X + 1}^X$$

state space is $Q = \mathcal{X} \cup \bigcup_{X \in \mathcal{X}} \text{sub}(\delta(X))$
Alternating Parity Krivine Automata

APKA of index $m$ is $A = (\mathcal{X}, \delta, I, \Lambda, (\tau_X)_{X \in \mathcal{X}})$ 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 $f_i^X : \tau_i^X$ for $i \leq n_X$ and $\varphi_X : \tau_X$.
- assignment of argument and value types

  $\tau_X = \tau_1^X \rightarrow \cdots \rightarrow \tau_n^X \rightarrow \tau_{n_X + 1}$

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

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

acceptance of an LTS by an APKA explained as 2-player game on configurations

\[ C = (s, (\psi, e), e', \Gamma, \Delta) \]

where

- \( s \) is current state in LTS

challenge: make fixpoint interaction in a play visible

Lemma: [[Bruse '18]] Every play can be re-arranged into a tree with a unique infinite path s.t. the outermost fixpoint on this path faithfully determines the winner of the play.
Environments and Closures

acceptance of an LTS by an APKA explained as 2-player game on configurations

\[ C = (s, (\psi, e), e', \Gamma, \Delta) \]

where

- \( s \) is current state in LTS
- \((\psi, e)\) current closure with \( \psi \in Q, \ e \in E \) environment binding variables to closures
- \( e' \) distinguished environment (point of current computation)
- \( \Gamma = (\psi_{n1}, e_{1n}), \ldots, (\psi_{11}, e_{11}) \) stack of closures
- \( \Delta \) stack of priorities

challenge: make fixpoint interaction in a play visible

**Lemma:** [[Bruse '18]] Every play can be re-arranged into a tree with a unique infinite path such that the outermost fixpoint on this path faithfully determines the winner of the play.
Environments and Closures

acceptance of an LTS by an APKA explained as 2-player game on configurations

\[ C = (s, (\psi, e), e', \Gamma, \Delta) \]

where

- \( s \) is current state in LTS
- \((\psi, e)\) current closure with \( \psi \in Q, e \in \mathcal{E} \) environment binding variables to closures
- \( e' \) distinguished environment (point of current computation)

challenge: make fixpoint interaction in a play visible

Lemma: [[Bruse '18]] Every play can be re-arranged into a tree with a unique infinite path s.t. the outermost fixpoint on this path faithfully determines the winner of the play.
Environments and Closures

acceptance of an LTS by an APKA explained as 2-player game on configurations

\[ C = (s, (\psi, e), e', \Gamma, \Delta) \]

where

- \( s \) is current state in LTS
- \( (\psi, e) \) current closure with \( \psi \in Q, e \in E \) environment binding variables to closures
- \( e' \) distinguished environment (point of current computation)
- \( \Gamma = (\psi_n, e_{i_n}), \ldots, (\psi_1, e_{i_1}) \) stack of closures

challenge: make fixpoint interaction in a play visible

**Lemma: [[Bruse '18]]** Every play can be re-arranged into a tree with a unique infinite path s.t. the outermost fixpoint on this path faithfully determines the winner of the play.
Environments and Closures

acceptance of an LTS by an APKA explained as 2-player game on configurations

\[ C = (s, (\psi, e), e', \Gamma, \Delta) \]

where

- \( s \) is current state in LTS
- \( (\psi, e) \) current closure with \( \psi \in Q, e \in \mathcal{E} \) environment binding variables to closures
- \( e' \) distinguished environment (point of current computation)
- \( \Gamma = (\psi_n, e_{i_n}), \ldots, (\psi_1, e_{i_1}) \) stack of closures
- \( \Delta \) stack of priorities

challenge: make fixpoint interaction in a play visible

**Lemma:** [[Bruse '18]] Every play can be re-arranged into a tree with a unique infinite path s.t. the outermost fixpoint on this path faithfully determines the winner of the play.
Theorem 8 (Bruse '18)

$HFL^k = \text{order-}k\ \text{APKA}$

can be seen as generalisation of

Theorem 9 (Emerson/Jutla '91)

$\mu$-calculus = alternating parity tree auomata

important for what follows:

the acceptance game for an order-1 APKA on a binary tree can be encoded as a binary tree again

$\leadsto$ strictness of fixpoint alternation
1 Motivation

2 Specifying Properties using Modal Fixpoint Logic
   - The Modal $\mu$-Calculus
   - Higher-Order Fixpoint Logic
   - Computational Complexity and Decidability
   - Automata, Logic, Games
   - Fixpoint Quantifier Alternation
   - Polyadic Higher-Order Fixpoint Logic

3 Future Work / Open Questions
Fixpoint Alternation in the $\mu$-Calculus

fixpoint alternation . . .

- by example: $\nu X.\mu Y.\Diamond(\neg \neg (p \land X) \lor Y)$
- intuitively: inner fixpoint formula depends on outer of different type
Fixpoint Alternation in the $\mu$-Calculus

fixpoint alternation . . .

- by example: $\nu X. \mu Y. \Diamond ((p \land X) \lor Y)$
- intuitively: inner fixpoint formula depends on outer of different type

fixpoint alternation is obstacle for specifying program properties:

- computationally: requires fixpoint iterations to be nested
- pragmatically: makes formulas harder to understand

but . . .

**Theorem 10** (Bradfield '96, Arnold '99, . . .)

*The alternation hierarchy in $\mathcal{L}_\mu$ is strict.*
Fixpoint Alternation in HFL

**Obs.:** according to “standard” def., every HFL formula is equivalent to an alternation-free one

**Ex.:** $\nu X.\mu Y. (p \land \Box X) \lor \Box Y \equiv \nu X. ((\lambda Z. \mu Y. (p \land \Box Z) \lor \Box Y) Z)$

もらえ fixpoint alternation hidden through higher types
Fixpoint Alternation in HFL

Obs.: according to “standard” def., every HFL formula is equivalent to an alternation-free one

Ex.: $\nu X. \mu Y.(p \land \diamond X) \lor \diamond Y \equiv \nu X.((\lambda Z. \mu Y.(p \land \diamond Z) \lor \diamond Y) Z)$

⇝ fixpoint alternation hidden through higher types

alternative suggestion: use automata-logic connection

Def.: alternation index of an HFL formula of order $k$ is the smallest number of priorities of an equivalent APKA of order $k$

Ex. (cont.): $\nu X. \mu Y.(p \land \diamond X)$ has equivalent APKA

• of order 0 with priorities $\{1, 2\}$
• of order 1 with priorities $\{0, 1\}$
higher-order does not conquer fixpoint alternation

**Theorem 11 (L. '02, Bruse '18)**

*The alternation hierarchy in HFL\(^1\) is strict.*

**Proof idea:** uses encoding of order-1 APKA run on binary tree as binary tree and Banach's Fixpoint Theorem, cmp. [Arnold, '99]

There are hard APKA \(\mathcal{A}_0\) that define acceptance:

\[
t \in L(\mathcal{A}) \quad \text{iff} \quad \text{run}(\mathcal{A}, t) \in L(\mathcal{A}_0)
\]

\(\leadsto L(\mathcal{A}_0)\) requires different fixpoint alternation
Fixpoint Alternation II: Collapses I

link to loss of small model property:

**Theorem 12** (*Bruse/L./Lozes '17*)

*The \( L_\mu \) fixpoint alternation hierarchy collapses over finite structures into alternation-free HFL\(^1\).*

**Proof:** use fact that on finite structures fixpoint iteration stops after finitely many steps

greatest fixpoint iteration can be expressed as a least fixpoint of order 1:

\[
\nu X.\varphi(X) \equiv (\mu F.\lambda X.(X \land \square^*(X \rightarrow \varphi(X))) \lor (F \varphi(X))) \lor "X \subseteq \varphi(X)"
\]
Fixpoint Alternation II: Collapses II

trick can be extended to order 1

**Theorem 13 (Bruse/L./Lozes ’17)**

The HFL$^1$ fixpoint alternation hierarchy **collapses over finite structures** into alternation-free HFL$^2$.

problem here: test whether greatest fixpoint of order 1 has been reached: “$\forall X : f(X) \subseteq \varphi(f)(X)$”
Fixpoint Alternation II: Collapses II

trick can be extended to order 1

Theorem 13 (Bruse/L./Lozes ’17)

The $HFL^1$ fixpoint alternation hierarchy collapses over finite structures into alternation-free $HFL^2$.

problem here: test whether greatest fixpoint of order 1 has been reached: “$\forall X : f(X) \subseteq \varphi(f)(X)$”

possible to enumerate all sets $X$ on linearly ordered structures but impossible on general structures due to bisimulation-invariance

observation: “$\forall$ modally definable $X : f(X) \subseteq \varphi(f)(X)$” suffices!

$$\nu H(t). (\bigwedge_{p \in P} t(p)) \land \bigwedge_{a \in A} H(\lambda x. t(\langle a \rangle x))$$

$$\land H(\lambda x. t(\neg x)) \land H(\lambda x. H(\lambda y. t(x \lor y)))$$
Fixpoint Alternation II: Collapse III

technique can be extended even further

note: order-$2$ function has order-$1$ functions as arguments

\[ \vdash \text{need to enumerate all functions of the form } \lambda x_1 \ldots \lambda x_m. \varphi \text{ with modal } \varphi \text{ when checking for termination of fixpoint iteration, e.g. for } m = 1: \]

\[
\nu H(t). ( \bigwedge_{p \in P} t(\lambda x. p)) \land t(\lambda x. x) \land H(\lambda f. t(\lambda x. \neg f(x))) \\
\land \bigwedge_{a \in A} H(\lambda f. t(\lambda x. \langle a \rangle f(x))) \land H(\lambda f_1. H(\lambda f_2. t(\lambda x. f_1(x) \lor f_2(x))))
\]
Fixpoint Alternation II: Collapse III

technique can be extended even further

note: order-2 function has order-1 functions as arguments

⇝ need to enumerate all functions of the form $\lambda x_1 \ldots \lambda x_m. \varphi$ with modal $\varphi$ when checking for termination of fixpoint iteration, e.g. for $m = 1$:

$$\nu H(t). (\bigwedge_{p \in P} t(\lambda x.p)) \land t(\lambda x.x) \land H(\lambda f.t(\lambda x. \neg f(x)))$$

$$\land \bigwedge_{a \in A} H(\lambda f.t(\lambda x. \langle a \rangle f(x))) \land H(\lambda f_1.H(\lambda f_2.t(\lambda x.f_1(x) \lor f_2(x))))$$

**Theorem 14** (Bruse/L./Lozes '17)

*The $HFL^2$ fixpoint alternation hierarchy collapses over finite structures into alternation-free $HFL^3$.***
1 Motivation

2 Specifying Properties using Modal Fixpoint Logic
   - The Modal $\mu$-Calculus
   - Higher-Order Fixpoint Logic
   - Computational Complexity and Decidability
   - Automata, Logic, Games
   - Fixpoint Quantifier Alternation
   - Polyadic Higher-Order Fixpoint Logic

3 Future Work / Open Questions
Polyadic Modal Logics

\(\mu\)-calculus and HFL (etc.) are monadic: they define a set of states in each TS

polyadic modal logics are interpreted in tuples \(\sim\) define relations of predetermined arity

syntactic solution: use tokens / names 1, 2, \ldots, \(r\)

classic example \([\text{Andersen '94; Otto '99}]

\[
\nu X. \left( \bigwedge_{p \in P} p(1) \rightarrow p(2) \right) \land \left( \bigwedge_{a \in \Sigma} [a]_1 \langle a \rangle_2 X \right) \land \{1 \leftrightarrow 2\} X
\]
Polyadic Modal Logics

μ-calculus and HFL (etc.) are monadic: they define a set of states in each TS

polyadic modal logics are interpreted in tuples \( \sim \) define relations of predetermined arity

syntactic solution: use tokens / names \( 1, 2, \ldots, r \)

classic example

\[
\varphi_{\text{bis}} := \nu X. \left( \bigwedge_{p \in P} p(1) \rightarrow p(2) \right) \land \left( \bigwedge_{a \in \Sigma} [a]_1\langle a\rangle_2 X \right) \land \{1 \leftrightarrow 2\} X
\]

defines bisimilarity \( \sim \); in general:

**Theorem 15 (Otto '99)**

\[\text{PHFL}^0 \equiv \text{PTIME}/\sim\]
Polyadic Higher-Order Fixpoint Logic

Polyadicity can be integrated into HFL $\leadsto$ PHFL

$$\varphi ::= p(i) \mid X \mid \varphi \lor \varphi \mid \neg \varphi \mid \langle a \rangle_i \varphi \mid \{ \kappa \} \varphi \mid \mu (X : \tau) . \varphi \mid \lambda (X^\tau : \tau) . \varphi \mid \varphi \varphi$$

with $1 \leq i \leq r$ and $\kappa : [r] \to [r]$ for some fixed arity $r \geq 1$

all other notions extend straight-forwardly with $\llbracket \text{Pr} \rrbracket = 2^{Sr}$
Polyadic Higher-Order Fixpoint Logic

Polyadicity can be integrated into HFL $\leadsto$ PHFL

$$\varphi ::= p(i) \mid X \mid \varphi \lor \varphi \mid \neg \varphi \mid \langle a \rangle_i \varphi \mid \{\kappa\} \varphi \mid \mu(X : \tau). \varphi \mid \lambda(X^\gamma : \tau). \varphi \mid \varphi \varphi$$

with $1 \leq i \leq r$ and $\kappa : [r] \rightarrow [r]$ for some fixed arity $r \geq 1$

all other notions extend straight-forwardly with $[\Pr] = 2^{Sr}$

Ex.: $(\nu F(X, Y). (X \rightarrow Y) \land \bigwedge_{a \in \Sigma} F(\langle a \rangle_1 X, \langle a \rangle_2 Y))$ fin(1) fin(2)
Polyadic Higher-Order Fixpoint Logic

Polyadicity can be integrated into HFL $\rightsquigarrow$ PHFL

$$\varphi ::= p(i) \mid X \mid \varphi \lor \varphi \mid \neg \varphi \mid \langle a \rangle_i \varphi \mid \{\kappa\}\varphi \mid \mu(X : \tau).\varphi \mid \lambda(X^\nu : \tau).\varphi \mid \varphi \varphi$$

with $1 \leq i \leq r$ and $\kappa : [r] \rightarrow [r]$ for some fixed arity $r \geq 1$

all other notions extend straight-forwardly with $\lceil \text{Pr} \rceil = 2^{Sr}$

Ex.: $(\nu F(X, Y).(X \rightarrow Y) \land \bigwedge_{a \in \Sigma} F(\langle a \rangle_1 X, \langle a \rangle_2 Y))$ fin(1) fin(2)

expresses NFA universality

note: PHFL$^1$ can express PSPACE-complete problems

what exactly is the expressive power of each PHFL$^k$?
Declarative Complexity Theory

$\text{PHFL}^k$ is a natural specification language for bisimulation-invariant properties

**Theorem 16 (L./Lozes ’14, Kronenberger ’18)**

- $\text{PHFL}^k \equiv k\text{-EXPTIME}/\sim$ for $k \geq 0$
Declarative Complexity Theory

PHFL$^k$ is a natural specification language for bisimulation-invariant properties

Theorem 16 (L./Lozes ’14, Kronenberger ’18)

(a) $\text{PHFL}^k \equiv k\text{-EXPTIME}/\sim$ for $k \geq 0$

(b) tail-recursive $\text{PHFL}^k \equiv (k - 1)\text{-EXPSPACE}/\sim$ for $k > 0$
Declarative Complexity Theory

$\text{PHFL}^k$ is a natural specification language for bisimulation-invariant properties

**Theorem 16** (L./Lozes ’14, Kronenberger ’18)

1. $\text{PHFL}^k \equiv k\text{-EXPTIME}/\sim$ for $k \geq 0$
2. tail-recursive $\text{PHFL}^k \equiv (k - 1)\text{-EXPSPACE}/\sim$ for $k > 0$

**Proof:** upper bounds by reduction of model checking problems from $\text{PHFL}^k$ to $\text{HFL}^k$

lower bounds with the help of intermediate logics using

1. $\text{HO}^{k+1} + \text{LFP} \equiv k\text{-EXPTIME}$ [Immerman ’87, Freire/Martins ’11]
2. $\text{HO}^{k+1} + \text{PFP} \equiv k\text{-EXPSPACE}$

[Abiteboul/Vianu ’87, Bruse/Kronenberger ’xx]
1 Motivation

2 Specifying Properties using Modal Fixpoint Logic
   - The Modal $\mu$-Calculus
   - Higher-Order Fixpoint Logic
   - Computational Complexity and Decidability
   - Automata, Logic, Games
   - Fixpoint Quantifier Alternation
   - Polyadic Higher-Order Fixpoint Logic

3 Future Work / Open Questions
Open Questions: Fixpoint Alternation Strictness

how do fixpoint alternation and type order interact in detail?

Conjecture: the fixpoint alternation hierarchy is strict within each \( \text{HFL}^k \) and even within HFL over the class of all structures / trees
Open Questions: Collapse Classes

collapse Theorems. 12–14 stated for class $\mathbb{T}_{\text{fin}}$ of finite structures can be strengthened clearly hold for class $\mathbb{T}_{\sim\text{fin}}$ of structures with finite bisimulation quotients even for classes of structures with finite closure ordinals
Open Questions: Collapse Classes

collapse Theorems. 12–14 stated for class $T_{\text{fin}}$ of finite structures can be strengthened clearly hold for class $T_{\text{fin}}^{\sim}$ of structures with finite bisimulation quotients even for classes of structures with finite closure ordinals

Conjecture: all inclusions in

$$T_{\text{fin}}^0 \supseteq T_{\text{fin}}^1 \supseteq \cdots \supseteq \bigcap_{k \in \mathbb{N}} T_{\text{fin}}^k \supseteq T_{\text{fin}}^{\sim} \supsetneq T_{\text{fin}}$$

are strict where $T_{\text{fin}}^k = \text{structures on which HFL}^k$-definable fixpoint iterations stabilise after finitely many steps
Open Questions: A Proof Theory

$\Sigma^1_1$-hardness makes axiomatisability a difficult question

**Open question:** Are there fragments of PHFL that can be axiomatised?

benefit: could reduce question after inclusion between program equivalences / pre-orders to finding proofs in PHFL

**Ex.:** $\vdash \varphi_{\text{bis}} \rightarrow \varphi_{\text{trace}}$?
The End