Specifying Program Properties Using Modal Fixpoint Logics

Martin Lange

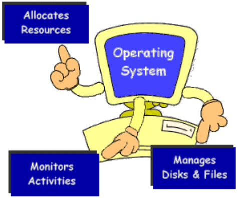
University of Kassel, Germany

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

Verification of Reactive Systems

general motivation: formal verification of dynamic systems

typical ICT systems are reactive:



Requirements for Specification Languages

generally needed for formal verification: formal specification languages, i.e. logics

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

- ... immediate behaviour: modal operators
  "*it is possible to react to any input of the form ...*"
  $\rightsquigarrow \Diamond\varphi, \Box\varphi$
- ... behaviour in the infinite: limit operators
  "*every request is eventually granted*"
  convenient tool: least and greatest fixpoints
  $\rightsquigarrow \mu X.\varphi(X), \nu X.\varphi(X)$

The Modal μ-Calculus

multi-modal logic + extremal fixpoint quantifiers

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

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

The Modal μ-Calculus

multi-modal logic + extremal fixpoint quantifiers

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

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

interpreted over transition system
$\mathcal{T} = (S, \{\xrightarrow{a} \mid a \in A\}, L : S \rightarrow 2^P)$

semantics usually given as $[\![\varphi]\!]^{\mathcal{T}}_{\rho} \subseteq S$ with Knaster-Tarski

Examples

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

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

Examples

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

- $\nu X.\langle a\rangle X$
- $\mu X.p \vee (\Diamond \mathtt{tt} \wedge \Box X)$ ($\equiv \mathtt{AF}p$ in CTL)

Examples

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

- $\nu X.\langle a\rangle X$
- $\mu X.p \lor (\Diamond\mathtt{tt} \land \Box X)$ ($\equiv$ $\mathtt{AF}p$ in CTL)
- $\mu X.[a]X$

Examples

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

- $\nu X.\langle a \rangle X$
- $\mu X.p \lor (\Diamond \mathtt{tt} \land \Box X)$ ($\equiv$ $\mathtt{AF}p$ in CTL)
- $\mu X.[a]X$
- $\nu X.\mu Y.\Diamond((p \land X) \lor Y)$

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 $\mathcal{L}_\mu$

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

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
- ...

Types

we need a simple type system with variances

$$\begin{aligned} \tau &::= \mathsf{Pr} \mid \tau^{v} \to \tau \\ v &::= + \mid - \mid 0 \end{aligned}$$

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

Types

we need a simple type system with variances

$$\begin{aligned} \tau &::= \mathsf{Pr} \mid \tau^v \to \tau \\ v &::= + \mid - \mid 0 \end{aligned}$$

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

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

$$V^+ := (M, \sqsubseteq) \qquad V^- := (M, \sqsupseteq) \qquad V^0 := (M, =)$$

Types

we need a simple type system with variances

$$\begin{aligned} \tau &::= \mathsf{Pr} \mid \tau^v \to \tau \\ v &::= + \mid - \mid 0 \end{aligned}$$

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

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

$$V^+ := (M, \sqsubseteq) \qquad V^- := (M, \sqsupseteq) \qquad V^0 := (M, =)$$

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

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

Formulas

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

[Viswanathan² '04]

$$\varphi \quad ::= \quad p \mid X \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle a \rangle \varphi \mid \mu\, X \quad .\varphi \mid \lambda\, X \quad .\varphi \mid \varphi\, \varphi$$

Formulas

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

[Viswanathan² '04]

$$\varphi \;::=\; p \mid X \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle a\rangle\varphi \mid \mu(X:\tau).\varphi \mid \lambda(X^{v}:\tau).\varphi \mid \varphi\,\varphi$$

well-formedness condition given by type system

needed to exclude $\langle a\rangle q\ \langle b\rangle p$, $\mu X.\neg X$, etc.

Formulas

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

[Viswanathan² '04]

$$\varphi \ ::= \ p \mid X \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle a \rangle \varphi \mid \mu(X : \tau).\varphi \mid \lambda(X^{v} : \tau).\varphi \mid \varphi\ \varphi$$

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.

$$\mu F(X, g).\neg X \vee F(g(X), g^2)$$

instead of

$$\mu(F : \mathrm{Pr}^- \to (\mathrm{Pr}^+ \to \mathrm{Pr})^+ \to \mathrm{Pr}).\lambda(X : \mathrm{Pr}).\lambda(g : \mathrm{Pr}^+ \to \mathrm{Pr}).$$
$$\neg X \vee F\ (g\ X)\ (\lambda(Y : \mathrm{Pr}^+).g\ (g\ Y))))$$

Negation is Trickier

why not simple condition as in the modal μ-calculus

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

e.g. $\neg\mu X.\neg\mu Y.\langle a\rangle\neg X \vee \langle b\rangle Y$

Negation is Trickier

why not simple condition as in the modal μ-calculus

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

e.g. $\neg\mu X.\neg\mu Y.\langle a\rangle\neg X \vee \langle b\rangle Y$

λ-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

φ well-formed iff $\emptyset \vdash \varphi : \mathsf{Pr}$ is derivable

$$\frac{}{\Gamma \vdash p : \mathsf{Pr}} \qquad \frac{v \in \{0,+\}}{\Gamma, X^v : \tau \vdash X : \tau} \qquad \frac{\Gamma^- \vdash \varphi : \mathsf{Pr}}{\Gamma \vdash \neg\varphi : \mathsf{Pr}}$$

$$\frac{\Gamma \vdash \varphi : \mathsf{Pr} \quad \Gamma \vdash \psi : \mathsf{Pr}}{\Gamma \vdash \varphi \vee \psi : \mathsf{Pr}} \qquad \frac{\Gamma \vdash \varphi : \mathsf{Pr}}{\Gamma \vdash \langle a \rangle \varphi : \mathsf{Pr}} \qquad \frac{\Gamma, X^v : \sigma \vdash \varphi : \tau}{\Gamma \vdash \lambda(X^v : \sigma).\varphi : (\sigma^v \to \tau)}$$

$$\frac{\Gamma \vdash \varphi : (\sigma^+ \to \tau) \quad \Gamma \vdash \psi : \sigma}{\Gamma \vdash (\varphi\ \psi) : \tau} \qquad \frac{\Gamma \vdash \varphi : (\sigma^- \to \tau) \quad \Gamma^- \vdash \psi : \sigma}{\Gamma \vdash (\varphi\ \psi) : \tau}$$

$$\frac{\Gamma \vdash \varphi : (\sigma^0 \to \tau) \quad \Gamma \vdash \psi : \sigma \quad \Gamma^- \vdash \psi : \sigma}{\Gamma \vdash (\varphi\ \psi) : \tau} \qquad \frac{\Gamma, X^+ : \tau \vdash \varphi : \tau}{\Gamma \vdash \mu(X : \tau).\varphi : \tau}$$

Semantics of HFL

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

$$
\begin{array}{rcl}
[\![\Gamma \vdash p : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} & = & \{s \in S \mid p \in L(s)\} \\
[\![\Gamma \vdash X : \tau]\!]^{\mathcal{T}}_{\eta} & = & \eta(X) \\
[\![\Gamma \vdash \neg\varphi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} & = & S \setminus [\![\Gamma^{-} \vdash \varphi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \neg\varphi : \sigma^{v} \to \tau]\!]^{\mathcal{T}}_{\eta} & = & f \in [\![\sigma^{v} \to \tau]\!] \text{ s.t. } \bar{f} = [\![\Gamma^{-} \vdash \varphi : \sigma^{v} \to \tau]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \varphi \vee \psi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} & = & [\![\Gamma \vdash \varphi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} \cup [\![\Gamma \vdash \psi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \langle a\rangle\varphi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} & = & \{s \in S \mid s \xrightarrow{a} t \text{ for some } t \in [\![\Gamma \vdash \varphi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta}\} \\
[\![\Gamma \vdash \lambda(X^{v} : \sigma).\varphi : \sigma^{v} \to \tau]\!]^{\mathcal{T}}_{\eta} & = & f \in [\![\sigma^{v} \to \tau]\!] \text{ s.t. } \forall x \in [\![\sigma]\!] \\
 & & \quad f\ x = [\![\Gamma, X^{v} : \sigma \vdash \varphi : \tau]\!]^{\mathcal{T}}_{\eta[X \mapsto x]} \\
[\![\Gamma \vdash \varphi\ \psi : \tau]\!]^{\mathcal{T}}_{\eta} & = & [\![\Gamma \vdash \varphi : \sigma^{v} \to \tau]\!]^{\mathcal{T}}_{\eta}\ [\![\Gamma' \vdash \psi : \sigma]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \mu(X : \tau)\varphi : \tau]\!]^{\mathcal{T}}_{\eta} & = & \bigsqcap\{x \in [\![\tau]\!] \mid [\![\Gamma, X^{+} : \tau \vdash \varphi : \tau]\!]^{\mathcal{T}}_{\eta[X \mapsto x]} \sqsubseteq_{\tau} x\}
\end{array}
$$

Semantics of HFL

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

$$
\begin{array}{rcl}
[\![\Gamma \vdash p : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} & = & \{s \in S \mid p \in L(s)\} \\
[\![\Gamma \vdash X : \tau]\!]^{\mathcal{T}}_{\eta} & = & \eta(X) \\
[\![\Gamma \vdash \neg\varphi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} & = & S \setminus [\![\Gamma^{-} \vdash \varphi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \neg\varphi : \sigma^{v} \rightarrow \tau]\!]^{\mathcal{T}}_{\eta} & = & f \in [\![\sigma^{v} \rightarrow \tau]\!] \text{ s.t. } \bar{f} = [\![\Gamma^{-} \vdash \varphi : \sigma^{v} \rightarrow \tau]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \varphi \vee \psi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} & = & [\![\Gamma \vdash \varphi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} \cup [\![\Gamma \vdash \psi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \langle a\rangle\varphi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta} & = & \{s \in S \mid s \xrightarrow{a} t \text{ for some } t \in [\![\Gamma \vdash \varphi : \mathrm{Pr}]\!]^{\mathcal{T}}_{\eta}\} \\
[\![\Gamma \vdash \lambda(X^{v} : \sigma).\varphi : \sigma^{v} \rightarrow \tau]\!]^{\mathcal{T}}_{\eta} & = & f \in [\![\sigma^{v} \rightarrow \tau]\!] \text{ s.t. } \forall x \in [\![\sigma]\!] \\
 & & \quad f\ x = [\![\Gamma, X^{v} : \sigma \vdash \varphi : \tau]\!]^{\mathcal{T}}_{\eta[X \mapsto x]} \\
[\![\Gamma \vdash \varphi\ \psi : \tau]\!]^{\mathcal{T}}_{\eta} & = & [\![\Gamma \vdash \varphi : \sigma^{v} \rightarrow \tau]\!]^{\mathcal{T}}_{\eta}\ [\![\Gamma' \vdash \psi : \sigma]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \mu(X : \tau)\varphi : \tau]\!]^{\mathcal{T}}_{\eta} & = & \bigsqcap\{x \in [\![\tau]\!] \mid [\![\Gamma, X^{+} : \tau \vdash \varphi : \tau]\!]^{\mathcal{T}}_{\eta[X \mapsto x]} \sqsubseteq_{\tau} x\}
\end{array}
$$

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?

$\big(\mu F(X).X \vee \langle a \rangle F(\langle b \rangle X)\big)\ \mathtt{tt}$

Examples

what properties are expressed by the following formulas?

$\big(\mu F(X).X \vee F(\Box X)\big)\ \mathtt{ff}$

Examples

what properties are expressed by the following formulas?

$$\varphi_{\mathsf{word}} := \neg \bigvee_{a \neq b} \big(\mu F(X,Y).\,(X \wedge Y) \vee F(\Diamond X, \Diamond Y)\big)\ \langle a \rangle \mathtt{tt}\ \langle b \rangle \mathtt{tt}$$

Examples

what properties are expressed by the following formulas?

$\big(\mu F(g, g', g''). \, (g \circ g' \circ g'') \vee F(g \circ \langle a \rangle, g' \circ \langle b \rangle, g'' \circ \langle c \rangle)\big) \; id \; id \; id \; \mathtt{tt}$

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?

$\big(\nu F(X).[b]X \wedge [a]F(F(X))\big)\ \mathtt{ff}$

Examples

what properties are expressed by the following formulas?

$$\big(\mu F(g).\, (g \circ g) \vee \bigvee_{a \in \Sigma} F(g \circ \langle a \rangle)\big)\ id\ \mathtt{tt}$$

Examples

what properties are expressed by the following formulas?

$\psi_m \, \psi_{m-1} \, \ldots \, \psi_1 \, \Diamond \, \Box \mathtt{ff}$ where $\psi_i := \lambda F.\lambda X.F\,(F\,X)$

Fragments by Type Order

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

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

Fragments by Type Order

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

$\mathsf{HFL}^{k,m}$ = 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: $ord(\tau_1 \to \ldots \to \tau_m \to \mathsf{Pr}) = \max\{1 + ord(\tau_i)\}$

$\mathrm{HFL}^{k,m}$ = 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: $ord(\tau_1 \to \ldots \to \tau_m \to \mathsf{Pr}) = \max\{1 + ord(\tau_i)\}$

$\mathrm{HFL}^{k,m}$ = 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 $2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$

Model Checking HFL

Theorem 2 (Axelsson/L./Somla '07)

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

PROOF SKETCH: (upper bounds) consider height of lattices $[\![\tau]\!]$:

$$height(\tau_1 \to \ldots \to \tau_m \to \mathrm{Pr}) \;=\; (n+1) \cdot \prod_{i=1}^{m} |[\![\tau_i]\!]|$$

with

$$|[\![\tau_1 \to \ldots \tau_m \to \mathrm{Pr}]\!]| \;=\; 2^{n \cdot \prod_{i=1}^{m} |[\![\tau_i]\!]|}$$

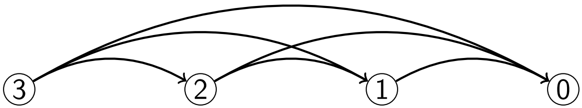
$\leadsto$ naïve bottom-up evaluation in time dominated by lattice height

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



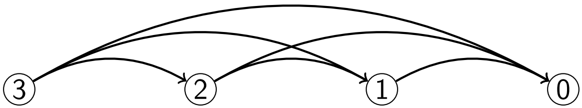
let $inc := \lambda X.X \leftrightarrow \Diamond\neg X$, what is $inc(\emptyset)$?

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



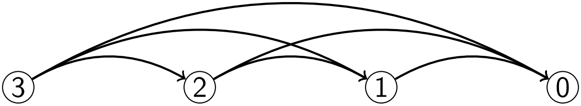
let $inc := \lambda X.X \leftrightarrow \Diamond\neg X$, what is $inc(\emptyset)$, $inc^k(\emptyset)$ for $k > 1$?

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

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

$\leadsto$ simulate run of space-bounded Turing machines

Tail Recursion

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

- not in both conjuncts $\rightsquigarrow$ no $\langle a \rangle X \langle b \rangle X$
- not behind modal box operators $\rightsquigarrow$ no $[a]X$
- not in argument position $\rightsquigarrow$ 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 $\rightsquigarrow$ no $\langle a \rangle X \langle b \rangle X$
- not behind modal box operators $\rightsquigarrow$ no $[a]X$
- not in argument position $\rightsquigarrow$ 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 HFL^1 is undecidable (at least Σ^1_1-hard)

follows from undecidability of Fixpoint Logic with Chop [Müller-Olm, '99] and embedding into HFL^1 [Viswanathan[2], '04]

undecidability not hard to see:

$$\varphi_{\mathsf{word}} \wedge \bigvee_{w \in L(G_1)} \langle w \rangle \mathtt{tt} \;\wedge \bigvee_{w \in L(G_2)} \langle w \rangle \mathtt{tt}$$

expresses non-emptiness of intersection between CFGs G_1 and G_2

No Finite Model Property

decidability of model checking and Σ^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:

$$(\mu X.\Box X) \land (\nu F(Y).Y \land F(\Diamond Y))\ \mathtt{tt}$$

forbids infinite paths but requires paths of unbounded length $\Box$

The Automata-Logic-Games Connection

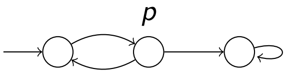
automata and games are important computational tools for temporal logics

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.:

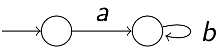


$$\stackrel{?}{\models} (\nu X.\mu Y.\Diamond((p \wedge X) \vee Y))$$

Stair-Parity Games

HFL model checking is not a parity game

Ex.: consider

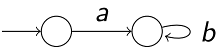


$$\stackrel{?}{\models} \big(\mu F(X).\langle b\rangle X \vee \langle a\rangle \nu G.F(G)\big)\ \mathtt{tt}$$

Stair-Parity Games

HFL model checking is not a parity game

Ex.: consider



$$\overset{?}{\models} \big(\mu F(X).\langle b\rangle X \vee \langle a\rangle \nu G.F(G)\big)\ \mathtt{tt}$$

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 ($\vee, \wedge, \langle a \rangle, [b]$)

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

challenge: get acceptance condition right, i.e. synchronise 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 $\mathcal{X} = \{X_1, \ldots, X_n\}$

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 $\Lambda : \mathcal{X} \to [1, m]$, resp. $[0, m-1]$

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 $\Lambda : \mathcal{X} \to [1, m]$, resp. $[0, m-1]$
- transition function $\delta : X \mapsto \varphi_X$, 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 $f_i^X : \tau_i^X$ for $i \leq n_X$ and $\varphi_X : \tau_X$.

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

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 $\Lambda : \mathcal{X} \to [1, m]$, resp. $[0, m-1]$
- transition function $\delta : X \mapsto \varphi_X$, 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 $f_i^X : \tau_i^X$ for $i \leq n_X$ and $\varphi_X : \tau_X$.
- assignment of argument and value types

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

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

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 $\Lambda : \mathcal{X} \to [1, m]$, resp. $[0, m-1]$
- transition function $\delta : X \mapsto \varphi_X$, 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 $f_i^X : \tau_i^X$ for $i \leq n_X$ and $\varphi_X : \tau_X$.

- assignment of argument and value types

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

- $I \in \mathcal{X}$ 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

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

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

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

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

where

- s is current state in LTS
- (ψ, e) current closure with $\psi \in \mathcal{Q}$, $e \in \mathcal{E}$ environment binding variables to 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
- (ψ, e) current closure with $\psi \in \mathcal{Q}$, $e \in \mathcal{E}$ 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
- (ψ, e) current closure with $\psi \in \mathcal{Q}$, $e \in \mathcal{E}$ 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

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

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

where

- s is current state in LTS
- (ψ, e) current closure with $\psi \in \mathcal{Q}$, $e \in \mathcal{E}$ 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
- Δ 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.

(Tree) Automata and Logics

Theorem 8 (Bruse '18)

HFL^k = order-k APKA

can be seen as generalisation of

Theorem 9 (Emerson/Jutla '91)

μ-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

$\rightsquigarrow$ strictness of fixpoint alternation

Fixpoint Alternation in the μ-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 in the μ-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.: $\nu X.\mu Y.(p \wedge \Diamond X) \vee \Diamond Y \equiv \nu X.\big((\lambda Z.\mu Y.(p \wedge \Diamond Z) \vee \Diamond Y)\ Z\big)$

$\leadsto$ 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.: $\nu X.\mu Y.(p \wedge \Diamond X) \vee \Diamond Y \equiv \nu X.\big((\lambda Z.\mu Y.(p \wedge \Diamond Z) \vee \Diamond Y)\ Z\big)$

$\rightsquigarrow$ 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.): $\nu X.\mu Y.(p \wedge \Diamond X)$ has equivalent APKA

- of order 0 with priorities $\{1, 2\}$
- of order 1 with priorities $\{0, 1\}$

Fixpoint Alternation I: Strictness

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:

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

$\leadsto$ $\overline{L(\mathcal{A}_0)}$ requires different fixpoint alternation

Fixpoint Alternation II: Collapses I

link to loss of small model property:

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

The $\mathcal{L}_\mu$ 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:

$$\nu X.\varphi(X) \equiv \left(\mu F.\lambda X.(X \land \underbrace{\Box^*(X \to \varphi(X)))}_{\text{“}X\subseteq\varphi(X)\text{”}} \lor (F\ \varphi(X))\right)\ \mathtt{tt}$$

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!

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

Fixpoint Alternation II: Collapse III

technique can be extended even further

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

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

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

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

Fixpoint Alternation II: Collapse III

technique can be extended even further

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

$\rightsquigarrow$ need to enumerate all functions of the form $\lambda x_1 \ldots \lambda x_m.\varphi$ with modal φ 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.

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 $\rightsquigarrow$ define relations of predetermined arity

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

classic example [Andersen '94; Otto '99]

$$\nu X.(\bigwedge_{p \in P} p(1) \to p(2)) \wedge (\bigwedge_{a \in \Sigma} [a]_1 \langle a \rangle_2 X) \wedge \{1 \leftrightarrow 2\} X$$

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 $\rightsquigarrow$ define relations of predetermined arity

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

classic example [Andersen '94; Otto '99]

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

defines bisimilarity $\sim$; in general:

Theorem 15 (Otto '99)

$PHFL^0 \equiv PTIME/{\sim}$

Polyadic Higher-Order Fixpoint Logic

polyadicity can be integrated into HFL $\rightsquigarrow$ PHFL

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

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 $[\![\mathrm{Pr}]\!] = 2^{S^r}$

Polyadic Higher-Order Fixpoint Logic

polyadicity can be integrated into HFL $\rightsquigarrow$ PHFL

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

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 $[\![\mathrm{Pr}]\!] = 2^{S^r}$

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

Polyadic Higher-Order Fixpoint Logic

polyadicity can be integrated into HFL $\rightsquigarrow$ PHFL

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

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 $[\![\mathrm{Pr}]\!] = 2^{S^r}$

Ex.: $(\nu F(X, Y).(X \to Y) \wedge \bigwedge_{a \in \Sigma} F(\langle a \rangle_1 X, \langle a \rangle_2 Y))\ \mathrm{fin}(1)\ \mathrm{fin}(2)$
expresses NFA universality

note: PHFL^1 can express PSPACE-complete problems

what exactly is the expressive power of each PHFL^k?

Declarative Complexity Theory

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

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

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

b) *tail-recursive PHFL^k $\equiv$ $(k-1)$-EXPSPACE$/\!\sim$ for $k > 0$*

Declarative Complexity Theory

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

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

a) *$PHFL^k \equiv k$-EXPTIME$/\!\sim$ for $k \geq 0$*
b) *tail-recursive $PHFL^k \equiv (k-1)$-EXPSPACE$/\!\sim$ for $k > 0$*

PROOF: upper bounds by reduction of model checking problems from PHFL^k to HFL^k

lower bounds with the help of intermediate logics using

a) HO^{k+1}+LFP $\equiv$ k-EXPTIME [Immerman '87, Freire/Martins '11]
b) HO^{k+1}+PFP $\equiv$ k-EXPSPACE
[Abiteboul/Vianu '87, Bruse/Kronenberger 'xx]

1. Motivation
2. Specifying Properties using Modal Fixpoint Logic
   - The Modal μ-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 HFL^k and even within HFL over the class of all structures / trees

Open Questions: Collapse Classes

collapse Theorems. 12–14 stated for class $\mathbb{T}_{\mathsf{fin}}$ of finite structures can be strengthened

clearly hold for class $\mathbb{T}^{\sim}_{\mathsf{fin}}$ 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 $\mathbb{T}_{\mathsf{fin}}$ of finite structures can be strengthened

clearly hold for class $\mathbb{T}_{\mathsf{fin}}^{\sim}$ of structures with finite bisimulation quotients

even for classes of structures with finite closure ordinals

Conjecture: all inclusions in

$$\mathbb{T}_{\mathsf{fin}}^{0} \supseteq \mathbb{T}_{\mathsf{fin}}^{1} \supseteq \cdots \supseteq \bigcap_{k\in\mathbb{N}} \mathbb{T}_{\mathsf{fin}}^{k} \supseteq \mathbb{T}_{\mathsf{fin}}^{\sim} \supsetneq \mathbb{T}_{\mathsf{fin}}$$

are strict where $\mathbb{T}_{\mathsf{fin}}^{k}$ = structures on which HFL^k-definable fixpoint iterations stabilise after finitely many steps

Open Questions: A Proof Theory

Σ^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.: $\vdash \varphi_{\mathsf{bis}} \rightarrow \varphi_{\mathsf{trace}}$?

The End