Functional Reachability

Nikos Tzevelekos
Oxford University Computing Laboratory

joint work with Luke Ong

Schloss Dagstuhl, June 2010
Reachability in functional computation.

- Consider a term $M$ of a higher-order functional programming language.
- Now consider a point $p$ inside $M$.
- Is there a program context $C$ such that computation of $C[M]$ reaches $p$?
Reachability in functional computation.

- Consider a term $M$ of a higher-order functional programming language.
- Now consider a point $p$ inside $M$.
- Is there a program context $C$ such that computation of $C[M]$ reaches $p$?
Reachability in functional computation.

- Consider a term $M$ of a higher-order functional programming language.
- Now consider a point $p$ inside $M$.
- Is there a program context $C$ such that computation of $C[M]$ reaches $p$?

Surprisingly, (Contextual) Reachability *per se* had not been studied in HO functional languages.

Idea: Use Games, Traversals, Automata.
Relevant work

- Control Flow Analysis.

*Compute at compile time the flow of control that is going to happen at run time.*
Relevant work

- Control Flow Analysis.

  Compute at compile time the flow of control that is going to happen at run time.

- In a HO-setting, the crucial element is that of closures.

- Reynolds (’70), Jones (’80), Shivers (’90), ... Malacaria & Hankin (late 90’s).

- CFA > Reach: more general.
  Reach > CFA: open vs closed world approach.
Relevant work

- Control Flow Analysis.

  *Compute at compile time the flow of control that is going to happen at run time.*

- In a HO-setting, the crucial element is that of *closures*.

- Reynolds (’70), Jones (’80), Shivers (’90), … Malacaria & Hankin (late 90’s).

- CFA > Reach: more general.
  Reach > CFA: open vs closed world approach.

- Useless code detection.

- Strictness analysis, etc.
The examined language: PCF.

- lambda-calculus,
- Boolean base type,
- recursion at all types.

\[
A, B ::= o \mid A \rightarrow B \\
v ::= t \mid f \\
M, N ::= v \mid x \mid \lambda x. M \mid MN \mid \text{if } M \text{ } N_1 \text{ } N_2 \mid Y_A
\]
\[ A, B ::= o \mid A \rightarrow B \]
\[ v ::= t \mid f \]
\[ M, N ::= v \mid x \mid \lambda x.M \mid MN \mid \text{if } M N_1 N_2 \mid Y_A \]
\[ A, B ::= o \mid A \to B \]
\[ v ::= t \mid f \]
\[ M, N ::= v \mid x \mid \lambda x.M \mid MN \mid \text{if } M\ N_1\ N_2 \mid \text{Y}_A \]

\[
(\lambda x.M)N \to M\{N/x\} \quad \text{if } t \to \lambda xy.x
\]
\[
\text{Y}M \to M(\text{Y}M) \quad \text{if } f \to \lambda xy.y
\]

\[ M \to N \implies E[M] \to E[N] \]

\[ E ::= [-] \mid E\ M \mid \text{if } E \]
A, B ::= o \mid A \to B

v ::= t \mid f

M, N ::= v \mid x \mid \lambda x. M \mid MN \mid \text{if } M N_1 N_2 \mid Y_A

- Write \((A_1, \ldots, A_n, o)\) for \(A_1 \to \cdots \to A_n \to o\).
- Divergence definable, e.g. \(\perp := Y_o(\lambda x.x)\).
\[ A, B ::= o \mid A \rightarrow B \]

\[ v ::= t \mid f \]

\[ M, N ::= v \mid x \mid \lambda x. M \mid MN \mid \text{if } M N_1 N_2 \mid Y_A \]

- Write \((A_1, \ldots, A_n, o)\) for \(A_1 \rightarrow \cdots \rightarrow A_n \rightarrow o\).
- Divergence definable, e.g. \(\perp ::= Y_o(\lambda x.x)\).
- \textbf{Finitary} restrictions (i.e. no \(Y\)):
  
  \(\text{fPCF} \quad M, N ::= v \mid x \mid \lambda x. M \mid MN \mid \text{if } M N_1 N_2\)
  
  \(\text{fPCF}_\perp \quad M, N ::= v \mid x \mid \lambda x. M \mid MN \mid \text{if } M N_1 N_2 \mid \perp\)
Reachability

- Given a closed PCF-term $M : (A_1, \ldots, A_n, o)$ and a coloured subterm $L$ of $M$,

- Are there closed PCF-terms $N_1 : A_1, \ldots, N_n : A_n$ and a coloured term $L'$ such that

$$MN \vec{\rightarrow} E[L']?$$
Reachability

- Given a closed PCF-term $M : (A_1, \ldots, A_n, o)$ and a *coloured* subterm $L$ of $M$,

- Are there closed PCF-terms $N_1 : A_1, \ldots, N_n : A_n$ and a coloured term $L'$ such that

$$M \vec{N} \rightarrow E[L']$$

*We can make things simpler*
PCF-with-error: PCF*

- Include an error constant: \( o = \{t, f, \star\} \)
- New rules: \( E[\star] \rightarrow \star \)
PCF-with-error: PCF$^*$

- Include an error constant: $o = \{t, f, \ast\}$
- New rules: $E[\ast] \rightarrow \ast$

$\ast$-Reachability:

- Given a closed PCF$^*$-term $M$ with exactly one $\ast$, 
- Are there closed PCF-terms $N_1, \ldots, N_n$ such that $M \vec{N} \rightarrow \ast$?
PCF-with-error: PCF\(^*\)

- Include an error constant: \( o = \{ t, f, \star \} \)
- New rules: \( E[\star] \rightarrow \star \)

Reachability \(\cong \star\)-Reachability

\(\star\)-Reachability:
- Given a closed PCF\(^*\)-term \( M \) with \textit{exactly one} \( \star \),
- Are there closed PCF-terms \( N_1, \ldots, N_n \) such that \( M \overrightarrow{N} \rightarrow \star \)?
$v$-REACH $[L_1, L_2]$:

- Given a closed $L_1$-term $M$, 
- Are there closed $L_2$-terms $N_1, \ldots, N_n$ such that $M \vec{N} \rightarrow v$?

E.g. $\star$-Reachability $= \star$-REACH $[\text{PCF}^{\dagger\star}, \text{PCF}]$
The REACH template is defined as follows:

\[ \nu\text{-REACH} [\mathcal{L}_1, \mathcal{L}_2] : \]

- Given a closed \( \mathcal{L}_1 \)-term \( M \),
- Are there closed \( \mathcal{L}_2 \)-terms \( N_1, \ldots, N_n \) such that \( M \overset{\vec{N}}{\rightarrow} \nu \)?

Where \( \nu \in \{t, f, \star\} \) and \( \mathcal{L}_1, \mathcal{L}_2 \subseteq \text{PCF}^* \).

Example:

\[ \begin{cases} 
\star\text{-Reachability} = \star\text{-REACH} [\text{PCF}^\star, \text{PCF}] \\
\nu\text{-REACH} [\mathcal{L}, \text{PCF}] = \nu\text{-REACH} [\mathcal{L}, \text{fPCF}] 
\end{cases} \]
\( v \text{-REACH}[L_1, L_2] \):

\[ v \in \{t, f, \star\} \text{ and } L_1, L_2 \subseteq \text{PCF}^* \]

- Given a closed \( L_1 \)-term \( M \),
- Are there closed \( L_2 \)-terms \( N_1, \ldots, N_n \) such that \( MN \rightarrow v \)?

E.g.
\[
\begin{align*}
\star\text{-Reachability} & = \star\text{-REACH}[\text{PCF}^\perp, \text{PCF}] \\
v\text{-REACH}[L, \text{PCF}] & = v\text{-REACH}[L, \text{fPCF}] 
\end{align*}
\]

From [Loader]:

Observational equivalence in \( \text{fPCF}_\perp \) is undecidable.

therefore:

\( t\text{-REACH}[\text{fPCF}_\perp, \text{fPCF}] \) is undecidable.
The following problems are undecidable.

- \texttt{t-REACH} \([\text{fPCF}_\bot, \text{fPCF}]\)
- \texttt{\star-REACH} \([\text{fPCF}^{\star\bot}, \text{fPCF}]\)
- \texttt{\star-Reachability}, i.e. \texttt{\star-REACH} \([\text{PCF}^{\star\bot}, \text{PCF}]\)
- Reachability
The following problems are undecidable.

- $t$-$REACH[fPCF_*^\bot, fPCF^*]$
- $\star$-$REACH[fPCF_{\bot^*}, fPCF^*]$
- $\star$-Reachability, i.e. $\star$-$REACH[PCF_{\bot^*}, PCF^*]$
- Reachability

*Not all is lost*

- $\star$-$REACH[fPCF_{\bot^*}^*, fPCF^*]$ ?
- Reachability for *finitary* $M$ ?
- $\star$-$REACH[fPCF_*^*, fPCF]$ ?
Our approach

We focus on $\nu$-REACH [$f\text{PCF}^*$, $f\text{PCF}$].

- Computations of $f\text{PCF}^*$-term $P : o$
- Traversals over its computation tree, $\lambda(P)$
- Runs of an Alternating Tree Automaton (ATA) on $\lambda(P)$
Our approach

We focus on $\nu$-\textsc{Reach} $[\text{fPCF}^*, \text{fPCF}]$.

Computations of $\text{fPCF}^*$-term $P : o$

Traversals over its computation tree, $\lambda(P)$

Runs of an \textit{Alternating Tree Automaton} (ATA) on $\lambda(P)$

$P \rightarrow \nu$ iff an ATA accepts $\lambda(P)$ on initial state with \textit{value} $\nu$. 
Starting from a $\text{fPCF}^*$-term $M$, 

- take its $\eta$-long form,
- add application symbols (@),
- view the result as a tree, $\lambda(M)$. 
Computation trees

Starting from a fPCF*-term $M$,

- take its $\eta$-long form,
- add application symbols (@),
- view the result as a tree, $\lambda(M)$.

$$(\lambda \Phi z. \Phi(\lambda y. \text{if } y \ast z)t) (\lambda f x. f x)t \quad \mapsto \quad \lambda \Phi z \quad \lambda f x \quad \lambda$$

$$\Phi \quad \lambda f x \quad \lambda$$

$$\lambda y \quad \lambda \quad \lambda$$

$$\text{if} \quad \lambda \quad \lambda$$

$$\lambda y \quad \lambda \quad \lambda$$

$$y \ast \quad z \quad \lambda$$
Starting from a fPCF*-term $M$,

- take its $\eta$-long form,
- add application symbols (@),
- view the result as a tree, $\lambda(M)$.

$$(\lambda \Phi z. \Phi(\lambda y. \text{if } y \star z)t) (\lambda f x. f x) t \quad \rightarrow \quad \lambda y \quad \lambda f x \quad \lambda t$$

For a finite fPCF*-alphabet $\Sigma$:

- $\lambda(M)$ is a $\Sigma$-labelled tree.
- $\lambda(M)$ is a $\Sigma$-labelled binding tree.
A traversal [Blum, Ong] over a *full* computation tree:

- follows the flow of control within it,
- seen from the perspective of Game Semantics.
Traversals

A traversal [Blum, Ong] over a full computation tree:

- follows the flow of control
- seen from the perspective of Game Semantics
A traversal [Blum, Ong] of a full computation tree:
- follows the flow of control
- seen from the perspective of Game Semantics
A traversal [Blum, Ong] over a full computation tree:

- follows the flow of control
- seen from the perspective of Game Semantics
A traversal [Blum, Ong] over a full computation tree:

- follows the flow of control
- seen from the perspective of Game Semantics
A traversal [Blum, Ong] over a full computation tree:

- follows the flow of control within it,
- seen from the perspective of Game Semantics.
A traversal [Blum, Ong] over a full computation tree:

- follows the flow of control within it,
- seen from the perspective of Game Semantics.
A **traversal** [Blum, Ong] over a full computation tree:

- follows the flow of control within it,
- seen from the perspective of Game Semantics.
A traversal [Blum, Ong] of a full computation tree:

- follows the flow of control within it,
- seen from the perspective of Game Semantics.
A **traversal** [Blum, Ong] of a full computation tree:

- follows the flow of control
- seen from the perspective of Game Semantics
A **traversal** [Blum, Ong] of a full computation tree:

- follows the flow of control within it,
- seen from the perspective of Game Semantics.
A traversal [Blum, Ong] over a full computation tree:

- follows the flow of control within it,
- seen from the perspective of Game Semantics.
Traversals

A traversal [Blum, Ong] over a full computation tree:

- follows the flow of control
- seen from the perspective of Game Semantics
A traversal [Blum, Ong] of a full computation tree:

- follows the flow of control within it,
- seen from the perspective of Game Semantics.
A traversal [Blum, Ong] over a full computation tree:

- follows the flow of control,
- seen from the perspective of Game Semantics.
Traversals

A **traversal** [Blum, Ong] over a full computation tree:

- follows the flow of control;
- seen from the perspective of Game Semantics.
A traversal [Blum, Ong] over a full computation tree:

- follows the flow of control,
- seen from the perspective of Game Semantics.
A traversal [Blum, Ong] of a full computation tree:

- follows the flow of control within it,
- seen from the perspective of Game Semantics.
A traversal [Blum, Ong] over a full computation tree:

- follows the flow of control within it,
- seen from the perspective of Game Semantics.

*-complete traversal
A traversal [Blum, Ong] over a full computation tree:

- follows the flow of control within it,
- seen from the perspective of Game Semantics.

A traversal is \( v \)-complete if:

- every question (red visit) has been answered (green visit),
- and the root question has been answered with \( v \).

For any \( P : o \) and \( v \), \( P \rightarrow v \) iff there is a \( v \)-complete traversal over \( \lambda(P) \).
An ATA is a quadruple $\mathcal{A} = \langle Q, \Sigma, q_0, \Delta \rangle$ where:

- $Q$ is a finite set of states,
- $\Sigma$ is a finite ranked alphabet,
- $q_0 \in Q$ is the initial state,
- $\Delta$ is a finite transition relation: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$ where $s \in \Sigma$, $q \in Q$, $Q_1, \ldots, Q_k \subseteq Q$. 

An ATA is a quadruple $\mathcal{A} = \langle Q, \Sigma, q_0, \Delta \rangle$ where:

- $Q$ is a finite set of states,
- $\Sigma$ is a finite ranked alphabet,
- $q_0 \in Q$ is the initial state,
- $\Delta$ is a finite transition relation: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.
An ATA is a quadruple $\mathcal{A} = \langle Q, \Sigma, q_0, \Delta \rangle$ where:

- $Q$ is a finite set of states,
- $\Sigma$ is a finite ranked alphabet,
- $q_0 \in Q$ is the initial state,
- $\Delta$ is a finite transition relation: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.
An ATA is a quadruple $\mathcal{A} = \langle Q, \Sigma, q_0, \Delta \rangle$ where:

- $Q$ is a finite set of states,
- $\Sigma$ is a finite ranked alphabet,
- $q_0 \in Q$ is the initial state,
- $\Delta$ is a finite transition relation: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.

\[ A(Q_1) \quad A(Q_2) \quad \ldots \quad A(Q_k) \]

\[ s_1 \quad s_2 \quad \ldots \quad s_k \]
How can we simulate a complete traversal by an ATA?
How can we simulate a complete traversal by an ATA?

- By *guessing* the number of visits of each node.
- By *guessing* the *profile* of each variable per visit.
- By verifying these guesses.
Variable profiles

Introduced by [Ong’06].

- \( \text{VP}_\Sigma(A_1, \ldots, A_n, o) := \text{Var}^A_\Sigma \times \text{Val} \times \mathcal{P}(\bigcup_{i=1}^n \text{VP}_\Sigma(A_i)) \)

- Notation: \((x, v), (x, v \mid \pi_1, \ldots, \pi_n)\)
Variable profiles

\[ \text{VP}(A_1, \ldots, A_n, o) := \text{Var} \times \text{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \text{VP}(A_i)) \]

Notation: \((x, v), (x, v | \pi_1, \ldots, \pi_n)\)
\[ \text{VP}(A_1, \ldots, A_n, o) := \text{Var} \times \text{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \text{VP}(A_i)) \]

Notation: \((x, v), (x, v \mid \pi_1, \ldots, \pi_n)\)
Variable profiles

$$\text{VP}(A_1, \ldots, A_n, o) := \text{Var} \times \text{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \text{VP}(A_i))$$

Notation: $$(x, v), (x, v | \pi_1, \ldots, \pi_n)$$
\( \text{VP}(A_1, \ldots, A_n, o) := \text{Var} \times \text{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \text{VP}(A_i)) \)

Notation: \((x, v), (x, v \mid \pi_1, \ldots, \pi_n)\)
Variable profiles

\[ \text{VP}(A_1, \ldots, A_n, o) := \text{Var} \times \text{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \text{VP}(A_i)) \]

Notation: \((x, v), (x, v \mid \pi_1, \ldots, \pi_n)\)
Variable profiles

\[ \text{VP}(A_1, \ldots, A_n, o) := \text{Var} \times \text{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \text{VP}(A_i)) \]

Notation: \((x, v), (x, v \mid \pi_1, \ldots, \pi_n)\)
Variable profiles

\[
\text{VP}(A_1, \ldots, A_n, o) := \text{Var} \times \text{Val} \times \mathcal{P}(\bigcup_{i=1}^{n} \text{VP}(A_i))
\]

Notation: \((x, v), (x, v \mid \pi_1, \ldots, \pi_n)\)
Given a finite fPCF*-alphabet $\Sigma$, the states of the \textit{traversal-simulating} ATA $A_\Sigma$ are:

$$Q := Val \times VP_\Sigma \times \mathcal{P}(VP_\Sigma)$$
Given a finite fPCF*-alphabet $\Sigma$, the states of the traversal-simulating ATA $A_\Sigma$ are:

$$Q := \text{Val} \times \mathcal{P}(\text{VP}_\Sigma) \times \mathcal{P}(\text{VP}_\Sigma)$$

$P \rightarrow v$ iff $A_\Sigma$ accepts $\lambda(P)$ on initial state with value $v$.

Any tree accepted by $A_\Sigma$ on closed initial state represents a closed fPCF*-term.
**Theorem:** $M : (A_1, \ldots, A_n, o) \in \nu$-REACH $[\text{fPCF}_\Sigma^*, \text{fPCF}_\Sigma]$ iff there is a closed initial state $q_0$ with value $\nu$ such that:

- $A_\Sigma(q_0)$ accepts $\lambda(M)$,
- $\forall i$, the language accepted by $A_{\Sigma}(q_0 \upharpoonright A_i)$ is non-empty.
Theorem: \( M : (A_1, \ldots, A_n, o) \in v\text{-REACH} [\text{fPCF}_\Sigma^*, \text{fPCF}_\Sigma] \) iff there is a closed initial state \( q_0 \) with value \( v \) such that:

- \( A_\Sigma(q_0) \) accepts \( \lambda(M) \),

- \( \forall i, \) the language accepted by \( A_\Sigma(q_0 \upharpoonright A_i) \) is non-empty.

Corollary: \( *\text{-REACH} [\text{fPCF}^*, \text{fPCF}(n)] \) is decidable.

Corollary: \( *\text{-REACH} [\text{fPCF}^*, \text{fPCF}] \) is decidable up to order 3.
**Theorem:** $M : (A_1, \ldots, A_n, o) \in \nu\text{-REACH} [\text{fPCF}_\Sigma^*, \text{fPCF}_\Sigma]$ iff there is a closed initial state $q_0$ with value $\nu$ such that:

- $A_\Sigma(q_0)$ accepts $\lambda(M),$
- $\forall i$, the language accepted by $A_\Sigma(q_0 \upharpoonright A_i)$ is non-empty.

**Corollary:** $\star\text{-REACH} [\text{fPCF}^*, \text{fPCF}(n)]$ is decidable.

**Corollary:** $\star\text{-REACH} [\text{fPCF}^*, \text{fPCF}]$ is decidable up to order 3.

- For the general case we use *Alternating Dependency Tree Automata* [Stirling’09].
- **Corollary:** Emptiness problem is undecidable for ADTA’s.
• A new kind of Reachability problems.
• Some undecidability results.
• Some technology from game semantics.
• Characterisation by ATA’s and ADTA’s.
• Some (relativised) decidability results.
A new kind of Reachability problems.

Some undecidability results.

Some technology from game semantics.

Characterisation by ATA’s and ADTA’s.

Some (relativised) decidability results.

Revisit (semantic) CFA?

Reachability through intersection types?

Conjecture: $\star$-REACH $[\text{fPCF}^*, \text{fPCF}]$?