Functional Reachability

Luke Ong   Nikos Tzevelekos

Oxford University Computing Laboratory

24th Symposium on Logic in Computer Science
Los Angeles, August 2009.
**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 the 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 the 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 the computation of $C[M]$ reaches $p$?

Surprisingly, (Contextual) Reachability per se had not been studied in HO functional languages.
Control Flow Analysis: Approximate at compile time the flow of control 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, etc.
The examined language: PCF

Types: \quad A, B ::= o \mid A \rightarrow B

Terms: \quad M, N ::= x \mid \lambda x.M \mid MN \mid t \mid f \mid \text{if } M N_1 N_2 \mid Y_A

Contexts: \quad C ::= \ldots
The examined language: PCF

Types: \[ A, B ::= o \mid A \to B \]

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

Contexts: \[ C ::= \ldots \]

Reductions: \[
\begin{align*}
(\lambda x.M)N & \to M\{N/x\} & \text{if } t & \to \lambda xy.x \\
Y M & \to M(Y M) & \text{if } f & \to \lambda xy.y
\end{align*}
\]

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

Ev. Contexts: \[ E ::= [\_] \mid E M \mid \text{if } E \]
The examined language: PCF

Types: \[ A, B \ ::= o \mid A \to B \]

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

Contexts: \[ C \ ::= \ldots \]

Reductions: Call-by-name \( \lambda \)-calculus + if + \( Y \)

- Write \((A_1, \ldots, A_n, o)\) for \(A_1 \to \cdots \to A_n \to o\).
- Divergence definable, e.g. \(\perp := Y_o(\lambda x.x)\).
- Finitary restrictions (i.e. no \( Y \)): \text{fPCF, fPCF}_\perp.
Given a PCF-term $M$ and a coloured subterm $L$ of $M$, is there a program context $C$ such that $C[M] \rightarrow E[L']$ with $L'$ coloured?
Reachability

- Given a PCF-term $M$ and a coloured subterm $L$ of $M$,
- Is there a program context $C$ such that $C[M] \rightarrow E[L']$ with $L'$ coloured?

Equivalently:

- Given a closed PCF-term $M : (A_1, \ldots, A_n, o)$ and a coloured subterm $L$ of $M$,
- Are there closed PCF-terms $N_1, \ldots, N_n$ such that

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

with $L'$ coloured?
Take base type $o = \{t, f, \star\}$ with $\star$ an error constant:

$$E[\star] \rightarrow \star$$

**$\star$-Reachability:**

- Given a closed PCF*-term $M : (A_1, ..., A_n, o)$ that has exactly one occurrence of $\star$,
  
- are there closed PCF-terms $N_1, ..., N_n$ such that $M \cdot N \rightarrow \star$?
Take base type $\tau = \{ t, f, \star \}$ with $\star$ an error constant:

$$E[\star] \longrightarrow \star$$

**$\star$-Reachability:**

- Given a closed PCF*-term $M : (A_1, \ldots, A_n, \tau)$ that has exactly one occurrence of $\star$,

- are there closed PCF-terms $N_1, \ldots, N_n$ such that $M \vec{N} \rightarrow \star$?

**Lemma:** Reachability $\cong \star$-Reachability.
For $v \in \{t, f, \star\}$ and $\mathcal{L}_1, \mathcal{L}_2 \subseteq \text{PCF}^*$:

$v$-\textsc{Reach} $[\mathcal{L}_1, \mathcal{L}_2]$: Given a closed $\mathcal{L}_1$-term $M : (A_1, \ldots, A_n, o)$, are there closed $\mathcal{L}_2$-terms $N_1, \ldots, N_n$ such that $M \overset{\vec{N}}{\rightarrow} v$?

E.g. $\star$-Reachability $= \star$-\textsc{Reach} $[\text{PCF}^{1\star}, \text{PCF}]$. 

Reach template
For $v \in \{t, f, \star\}$ and $\mathcal{L}_1, \mathcal{L}_2 \subseteq \text{PCF}^*$:

$v$-$\mathbf{REACH} [\mathcal{L}_1, \mathcal{L}_2]$: 
Given a closed $\mathcal{L}_1$-term $M : (A_1, \ldots, A_n, o)$, are there closed $\mathcal{L}_2$-terms $N_1, \ldots, N_n$ such that $M \vec{N} \rightarrow v$?

Three classes of problems:

- Reachability
- $\star$-Reachability
- $\star$-$\mathbf{REACH} [\text{PCF}^{1*}, \text{PCF}]$
- $\star$-$\mathbf{REACH} [\text{PCF}^{1*}, \text{fPCF}]$
- $\star$-$\mathbf{REACH} [\text{fPCF}^{1*}, \text{fPCF}]$
- $\star$-$\mathbf{REACH} [\text{fPCF}^{1*}, \perp]$
An undecidability result

Lemma: $\ast\text{-REACH}[f\text{PCF}_\perp, f\text{PCF}]$ is undecidable.

Proof: By reduction of solvability of systems of $f\text{PCF}_\perp$-equations (proved undecidable by [Loader’01]).
Our approach

- We focus on $v$-REACH [fPCF*, fPCF].
- For fPCF*-term $P : o$,

  Computations of $P$ \[\text{Traversals over its computation tree, } \lambda(P)\]

  Runs of an Alternating Tree Automaton (ATA) on $\lambda(P)$
Our approach

- We focus on $v$-\textsc{Reach} [$fPCF^*, fPCF$].
- For $fPCF^*$-term $P : o$,

\begin{align*}
\text{Computations of } P & \xrightarrow{\text{Traversals over its computation tree, } \lambda(P)} \\
\text{Runs of an Alternating Tree Automaton (ATA) on } \lambda(P) & \xrightarrow{\text{if an ATA accepts } \lambda(P) \text{ on initial state with value } v.}
\end{align*}

$P \xrightarrow{v} \iff$ iff an ATA accepts $\lambda(P)$ on initial state with value $v$. 
Starting from a fPCF*-term $M$,

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

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

$$(\lambda \Phi z. \Phi(\lambda y. if y \ast z)t)(\lambda \varphi x. \varphi x)t \mapsto$$
A traversal [Blum, Ong] over a computation tree,

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

\[
\begin{align*}
\lambda & \Phi z \\
\Phi & \varphi x \\
\lambda y & t \\
\lambda & x \\
y & * \\
z & \\
\end{align*}
\]
A traversal [Blum, Ong] over a computation tree,
- follows the flow of control within it,
- seen from the perspective of Game Semantics.

\[
\lambda \Phi z \quad \lambda \varphi x \quad \lambda
\]
\[
\Phi \quad \varphi \quad t
\]
\[
\lambda y \quad \lambda \quad \lambda
\]
\[
\text{if} \quad t \quad x
\]
\[
\lambda \quad \lambda \quad \lambda
\]
\[
y \quad * \quad z
\]
A traversal [Blum, Ong] over a computation tree follows the flow of control within it, seen from the perspective of Game Semantics.
A traversal [Blum, Ong] over a computation tree, follows the flow of control within it, seen from the perspective of Game Semantics.
A traversal [Blum, Ong] over a computation tree follows the flow of control within it, seen from the perspective of Game Semantics.
A traversal [Blum, Ong] over a computation tree

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

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

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

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

- follows the flow of control within it,

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

\[ \lambda \Phi z \]

\[ \lambda \varphi x \]

\[ \lambda \]

\[ \lambda \]

\[ t \]

\[ x \]

\[ y \]

\[ z \]

\[ \text{*-complete traversal} \]
A traversal [Blum, Ong] over a computation tree,

■ follows the flow of control within it,

■ seen from the perspective of Game Semantics.

A traversal is \( v \text{-complete} \) if every question (red visit) has been answered (green visit), and the root question has been answered with \( v \).

**Theorem:** For any \( P : o \) and value \( v \), \( P \rightarrow v \) iff there is a complete \( v \)-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: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$. 

$s \in \Sigma$
$q \in Q$

$Q_1, \ldots, Q_k \subseteq Q$
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: $q \xrightarrow{s} (Q_1, \ldots, Q_k)$.
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].

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

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

\[ \lambda \Phi z \]

\[ \Phi \lambda \varphi x \]

\[ \varphi \lambda y \]

\[ \text{if} \]

\[ (y, t) \]

\[ (x, t) \]
Variable profiles
\[ (\Phi, \star \mid (\varphi, \star \mid (y, t)), (x, t)) \]

\[ (\varphi, \star \mid (y, t)) \]

\[ \text{if} \]

\[ (y, t) \]

\[ y \]

\[ \star \]

\[ z \]
Given a finite fPCF*-alphabet $\Sigma$, the states of the traversal-simulating ATA $A_\Sigma$ are:

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

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

- $M \vec{N} \rightarrow v$ iff $A_\Sigma$ accepts $\lambda(M \vec{N})$ on initial state with value $v$.
- Any tree accepted by $\tilde{A}_\Sigma$ is a closed fPCF-term.
**Theorem:** \( M \in v\text{-}\text{REACH}[fPCF^*_{\Sigma}, fPCF_{\Sigma}] \) iff there is an initial state \( q_0 \) with value \( v \) such that:

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

- \( \forall i, \) the language accepted by \( \tilde{A}_{\Sigma}(q_0 \upharpoonright A_i) \) is non-empty.
The Problem

Relevant work

The examined language: PCF

Reachability PCF-with-error: PCF

⋆ Reach template

An undecidability result

Our approach

Computation trees

Traversals

Alternating Tree Automata

Traversal-simulating ATA’s

Variable profiles

ATA correspondence

Results

Theorem: $M \in v\text{-REACH} \left[ \text{fPCF}^*, \text{fPCF}_\Sigma \right]$ iff there is an initial state $q_0$ with value $v$ such that:

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

- $\forall i$, the language accepted by $\tilde{A}_\Sigma(q_0 \upharpoonright A_i)$ is non-empty.

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

Corollary: $\ast\text{-REACH} \left[ \text{fPCF}^*, \text{fPCF} \right]$ is decidable up to order 3.
For the general case we can use Alternating Dependency Tree Automata [Stirling’09].

*Corollary:* Emptiness problem is undecidable for ADTA’s.
Conclusion and on

- 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.
Conclusion and on

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

THANKS!