Overview
Intuition of CSHORe
HorSat(T)

CSHORe and HorsSat(T): Saturation based higher-order model checking in practice

Christopher Broadbent
CSHORe: Arnaud Carayol, Matthew Hague and Olivier Serre
HorsSat(T): Naoki Kobayashi

12th July 2013
PUMA Seminar, LMU
Outline

1. Overview
2. Intuition of CSHORe
3. HorSat(T)
1 Overview

2 Intuition of CSHORE

3 HorSat(T)
Higher-Order Recursion Schemes (HORS)

Example

Terminals:  $f, a$

Non-Terminals:  $S, F$

$S \rightarrow F(fa)$

$F\phi \rightarrow \phi(F\phi)$

$S : o$

$F : (o \rightarrow o) \rightarrow o$

This is an order-2 scheme
Intuition of CSHORe

Higher-Order Recursion Schemes (HORS)

Example

Terminals: \( f, a \)  
Non-Terminals: \( S, F \)

\[
\begin{align*}
S & \rightarrow F(fa) \\
F\phi & \rightarrow \phi(F\phi)
\end{align*}
\]

\[
\begin{align*}
S &: o \\
F &: (o \rightarrow o) \rightarrow o
\end{align*}
\]

This is an order-2 scheme
Higher-Order Recursion Schemes (HORS)

Example

Terminals: \( f, a \)  
Non-Terminals: \( S, F \)

\[
S \rightarrow F(fa) \\
F\phi \rightarrow \phi(F\phi)
\]

\[
S : o \\
F : (o \rightarrow o) \rightarrow o
\]

This is an order-2 scheme
Higher-Order Recursion Schemes (HORS)

Example

Terminals: $f, a$  
Non-Terminals: $S, F$

$$
S \rightarrow F(fa) \\
F\phi \rightarrow \phi(F\phi)
$$

$S : o$

$F : (o \rightarrow o) \rightarrow o$

This is an order-2 scheme
Higher-Order Recursion Schemes (HORS)

Example

Terminals: $f, a$  Non-Terminals: $S, F$

\[
S \rightarrow F(fa) \\
F\phi \rightarrow \phi(F\phi)
\]

$S : o$

$F : (o \rightarrow o) \rightarrow o$

This is an order-2 scheme
Higher-Order Recursion Schemes (HORS)

Example

Terminals: $f, a$  Non-Terminals: $S, F$

- $S \rightarrow F(fa)$
- $F\phi \rightarrow \phi(F\phi)$

- $S : o$
- $F : (o \rightarrow o) \rightarrow o$

This is an order-2 scheme
The Recursion Scheme Generates an Infinite Tree

Example

\[ S \rightarrow F(fa) \]
\[ F\phi \rightarrow \phi(F\phi) \]

Term being evaluated:
The Recursion Scheme Generates an Infinite Tree

Example

\[ S \rightarrow F(fa) \]
\[ F\phi \rightarrow \phi(F\phi) \]

Term being evaluated: \( S \)

\( S \)
The Recursion Scheme Generates an Infinite Tree

Example

\[ S \rightarrow F(fa) \]
\[ F\phi \rightarrow \phi(F\phi) \]

Term being evaluated: \( F(fa) \)

\( F(fa) \)
The Recursion Scheme Generates an Infinite Tree

**Example**

\[ S \rightarrow F(fa) \]
\[ F\phi \rightarrow \phi(F\phi) \]

Term being evaluated: \( fa(F(fa)) \)
The Recursion Scheme Generates an Infinite Tree

Example

\[ S \rightarrow F(fa) \]
\[ F\phi \rightarrow \phi(F\phi) \]

Term being evaluated: \( fa(fa(F(fa))) \)
The Recursion Scheme Generates an Infinite Tree

Example

\[
S \rightarrow F(fa) \\
F\phi \rightarrow \phi(F\phi)
\]

Term being evaluated: \(fa(fa(fa(F(fa))))\)
The Recursion Scheme Generates an Infinite Tree

Example

\[ S \rightarrow F(fa) \]
\[ F\phi \rightarrow \phi(F\phi) \]

Term being evaluated: \( fa(fa(fa(\cdots))) \)
Given a finite automaton $\mathcal{A}$ and a HORS $\mathcal{G}$, does $\mathcal{A}$ have a possibly infinite run on $\llbracket \mathcal{G} \rrbracket$?
Applications

- MoCHi: Refinement type inference [Kobayashi et al., 2011, 2013]
- Safety checking of (networks of) HO tree transducers [Kobayashi et al. 2010, Unno et al. 2010]
- Safety checking of HORS + pattern matching [Ramsay and Ong 2011]
- Compressed data [Kobayashi et al. 2012]
- Exact flow analysis [Tobita et al. 2012]
Given a finite automaton $A$ and a HORS $\mathcal{G}$, does $A$ have a possibly infinite run on $\llbracket \mathcal{G} \rrbracket$?

**Theorem (Kobayashi, Ong, 2011)**

\textit{This problem is $(n - 1)$-EXPTIME complete.}
Given a finite automaton $\mathcal{A}$ and a HORS $\mathcal{G}$, does $\mathcal{A}$ have a possibly infinite run on $\llbracket \mathcal{G} \rrbracket$?

**Theorem (Kobayashi, Ong, 2011)**

*This problem is $(n - 1)$-EXPTIME complete.*
Consider the **product** of the HORS and property automaton.

Error configurations are those where the product system gets stuck.

Can one unfold \((q_0, S)\) without hitting such an error?
Consider the **product** of the HORS and property automaton.

**Error** configurations are those where the product system gets stuck.

Can one unfold \((q_0, S)\) without hitting such an **error**?
Consider the product of the HORS and property automaton.

Error configurations are those where the product system gets stuck.

Can one unfold \((q_0, S)\) without hitting such an error?
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Naïve Algorithm
Kobayashi’s Practical Methodology

**Idea:** Start from a much smaller set

1. Analyse the behaviour of the HORS-Automaton product in a **forward** direction to...
2. ...over-approximate the set of reachable configurations
3. Whittle this down to a maximal **forward-closed** subset.
Idea: Start from a much smaller set

1. Analyse the behaviour of the HORS-Automaton product in a forward direction to...
2. ...over-approximate the set of reachable configurations
3. Whittle this down to a maximal forward-closed subset.
Idea: Start from a much smaller set

1. Analyse the behaviour of the HORS-Automaton product in a **forward** direction to...
2. ...over-approximate the set of reachable configurations
3. Whittle this down to a maximal **forward-closed** subset.
Kobayashi’s Practical Methodology

Idea: Start from a much smaller set

1. Analyse the behaviour of the HORS-Automaton product in a forward direction to...
2. ...over-approximate the set of reachable configurations
3. Whittle this down to a maximal forward-closed subset.
Higher-Order Model-Checkers

- TReCS
- GTReCS
Higher-Order Model-Checkers

Naoki Kobayashi

TReCS

GTReCS
Overview
Intuition of CSHORe
HorSat(T)

Higher-Order Model-Checkers

Naoki Kobayashi
TReCS

GTRReCS

TravMC
Overview

Intuition of CSHORe

HorSat(T)

Higher-Order Model-Checkers

Naoki Kobayashi

TReCS

GTReCS

Robin Featherway et al.

TravMC

Broadbent CSHORe: Carayol, Hague, Serre
HorsSat: Kobayashi
Higher-Order Model-Checkers

TReCS

GTReCS

TravMC
Higher-Order Model-Checkers

- TReCS
- GTReCS
- TravMC

Game Semantics
Overview
Intuition of CSHORe
HorSat(T)

Higher-Order Model-Checkers

- TReCS
- GTReCS
- Concrete Unfolding
- Game Semantics
- TravMC
The TReCS and GTReCS Algorithms

Initial

Error
The TReCS and GTReCS Algorithms

Overview
Intuition of CSHORe
HorSat(T)
The TReCS and GTReCS Algorithms
The TReCS and GTReCS Algorithms
The TReCS and GTReCS Algorithms

Initial

Error
The TReCS and GTReCS Algorithms
The TReCS and GTReCS Algorithms
The TReCS and GTRReCS Algorithms
The TReCS and GTReCS Algorithms
The TReCS and GTReCS Algorithms
The TReCS and GTReCS Algorithms
The TReCS and GTReCS Algorithms
The TReCS and GTRReCS Algorithms
The TReCS and GTReCS Algorithms
The TReCS and GTReCS Algorithms
The TReCS and GTReCS Algorithms
The TReCS and GTReCS Algorithms
The TReCS and GTReCS Algorithms
The TReCS and GTRReCS Algorithms
The TReCS and GTRReCS Algorithms
The TReCS and GTReCS Algorithms

Initial

Error
The TReCS and GTRReCS Algorithms
The TReCS and GTReCS Algorithms
Higher-Order Model-Checkers

- TReCS
- GTReCS
- TravMC

Concrete Unfolding
Game Semantics
Overview
Intuition of CSHORe
HorSat(T)

Higher-Order Model-Checkers

TReCS

Forward Propagation

Concrete Unfolding

Game Semantics

GTReCS

TravMC
Overview
Intuition of CSHORe
HorSat(T)

Drawbacks Of Forward Propagation

Concrete Unfolding: Can take a long time to discover all relevant behaviours [related to pumping]
  - TReCS and TravMC may take \((n - 1)\)-EXPTIME in size of HORS (under suitable assumptions).

Abstract Unfolding: Can accumulate a lot of ‘incorrect’ types
  - Although GTReCS theoretically runs in linear time in size of HORS (under suitable assumptions).
Concrete Unfolding: Can take a long time to discover all relevant behaviours [related to pumping]
  - TReCS and TravMC may take \((n - 1)^{-\text{EXPTIME}}\) in size of HORS (under suitable assumptions).

Abstract Unfolding: Can accumulate a lot of ‘incorrect’ types
  - Although GTReCS theoretically runs in linear time in size of HORS (under suitable assumptions).
Algorithms that Work Forwards
Algorithms that Work Forwards

Initial
The Inherent ‘Spill-Over’
The Inherent ‘Spill-Over’
Irregularity of Reachable Terms

\[
\begin{align*}
S : & \quad o \\
F : & \quad (o \to o) \to o \\
G, H : & \quad o \to o \to o \\
S & \quad \to F H a \\
F \phi x & \quad \to F (G(\phi x)) x \\
F (H(\phi x)) x & \quad \mid \phi(\phi x)
\end{align*}
\]
Irregularity of Reachable Terms

\[ S : o \quad F : (o \rightarrow o) \rightarrow o \quad G, H : o \rightarrow o \rightarrow o \]

\[ S \quad \rightarrow \quad F H a \]

\[ F \phi x \quad \rightarrow \quad F (G(\phi x)) x \quad | \quad F (H(\phi x)) x \quad | \quad \phi(\phi x) \]
Overview
Intuition of CSHORE
HorSat(T)
Irregularity of Reachable Terms

\[ S : o \quad F : (o \rightarrow o) \rightarrow o \quad G, H : o \rightarrow o \rightarrow o \]

\[ S \rightarrow FHa \]

\[ F \phi x \rightarrow F(G(\phi x)) x \quad | \quad F(H(\phi x)) x \quad | \quad \phi(\phi x) \]
Irregularity of Reachable Terms

\[ S : o \quad F : (o \rightarrow o) \rightarrow o \quad G, H : o \rightarrow o \rightarrow o \]

\[ S \rightarrow FHa \]

\[ F\phi x \rightarrow F(G(\phi x))x \quad | \quad F(H(\phi x))x \quad | \quad \phi(\phi x) \]
Overview
Intuition of CSHORe
HorSat(T)

Irregularity of Reachable Terms

\[ S : o \quad F : (o \rightarrow o) \rightarrow o \quad G, H : o \rightarrow o \rightarrow o \]

\[ S \rightarrow FHa \]

\[ F \phi x \rightarrow F(G(\phi x))x \quad \text{and} \quad F(H(\phi x))x \quad \text{and} \quad \phi(\phi x) \]
Irregularity of Reachable Terms

\[ S : o \quad F : (o \rightarrow o) \rightarrow o \quad G, H : o \rightarrow o \rightarrow o \]

\[ S \rightarrow F H a \]

\[ F \phi x \rightarrow F (G(\phi x)) x \quad | \quad F (H(\phi x)) x \quad | \quad \phi(\phi x) \]
Irregularity of Reachable Terms

\[ S : o \quad F : (o \rightarrow o) \rightarrow o \quad G, H : o \rightarrow o \rightarrow o \]

\[ S \quad \rightarrow \quad F \quad H \quad a \]

\[ F \phi x \quad \rightarrow \quad F \left( G(\phi x) \right) x \quad | \quad F \left( H(\phi x) \right) x \quad | \quad \phi(\phi x) \]
Irregularity of Reachable Terms

\[ S : o \quad F : (o \rightarrow o) \rightarrow o \quad G, H : o \rightarrow o \rightarrow o \]

\[ S \rightarrow F H a \]

\[ F \phi x \rightarrow F (G(\phi x)) x \quad | \quad F (H(\phi x)) x \quad | \quad \phi(\phi x) \]
Irregularity of Reachable Terms

\[ S : o \quad F : (o \to o) \to o \quad G, H : o \to o \to o \]

\[ S \to F H a \]

\[ F \phi x \to F(G(\phi x)) x \quad | \quad F(H(\phi x)) x \quad | \quad \phi(\phi x) \]
Irregularity of Reachable Terms

\[ S : o \quad F : (o \rightarrow o) \rightarrow o \quad G, H : o \rightarrow o \rightarrow o \]

\[ S \rightarrow F H a \]

\[ F \phi x \rightarrow F (G(\phi x)) x \quad | \quad F (H(\phi x)) x \quad | \quad \phi(\phi x) \]
Irregularity of Reachable Terms

$S : o$  $F : (o \rightarrow o) \rightarrow o$  $G, H : o \rightarrow o \rightarrow o$

$S \rightarrow FHa$

$F \phi x \rightarrow F (G(\phi x)) x$  $|  F (H(\phi x)) x$  $|  \phi(\phi x)$
Irregularity of Reachable Terms

\[ S : o \quad F : (o \rightarrow o) \rightarrow o \quad G, H : o \rightarrow o \rightarrow o \]

\[ S \rightarrow F \, H \, a \]

\[ F \phi x \rightarrow F (G(\phi x)) x \quad | \quad F (H(\phi x)) x \quad | \quad \phi(\phi x) \]
Higher-Order Model-Checkers

- TReCS
- GTReCS
- TravMC

Forward Propagation

Concrete Unfolding

Game Semantics
Overview
Intuition of CSHORe
HorSat(T)

Higher-Order Model-Checkers

TReCS  \( \text{Forward} \)  GTReCS

Concrete Unfolding  \( \text{Game Semantics} \)

CSHORE  TravMC  HorSat

Broadbent CSHORE: Carayol, Hague, Serre HorsSat: Kobayashi  HorSat
Overview
Intuition of CSHORe
HorSat(T)

Higher-Order Model-Checkers

TReCS

HorSat

Forward Propagation

Concrete Unfolding

Game Semantics

with Arnaud Carayol, Matthew Hague and Olivier Serre

HorSat

CSHORe

with Arnaud Carayol, Matthew Hague and Olivier Serre
Overview

Intuition of CSHORe

HorSat(T)

Higher-Order Model-Checkers

TReCS

GTReCS

Forward Propagation

Concrete Unfolding

Game Semantics

with Arnaud Carayol, Matthew Hague and Olivier Serre

with Naoki Kobayashi

HorSat

CSHORe

Broadbent CSHORe: Carayol, Hague, Serre

HorsSat: Kobayashi
Higher-Order Model-Checkers

- TReCS
- GTReCS
- TravMC
- CSHORE
- HorSat

Forward Propagation
Concrete Unfolding
Game Semantics
1 Overview

2 Intuition of CSHORE

3 HorSat(T)
Naïve Saturation

Initial

Error
Naïve Saturation

- Initial
- Error
Naïve Saturation

Overview
Intuition of CSHORE
HorSat(T)

Initial Error

Broadbent CSHORE: Carayol, Hague, Serre HorsSat: Kobayashi HorSat
Naïve Saturation

Overview
Intuition of CSHORe
HorSat(T)
Naïve Saturation

Initial

Error
Naïve Saturation
Saturation for Pushdown Automata

At order-1 something along these lines works great...

...Moped [Schwoon and Esparza]
...jMoped [Suwimonteerabuth, Scwhoon, Berger and Esparza]
[Bouajjani et al. 1997], [Finkel et al. 1997]

Take a 1-PDA $\mathcal{C}$ with control-states $Q$ and alphabet $\Gamma$.

A $\mathcal{C}$-stack automaton $A$ has control-states $Q$ with $Q \subseteq Q$.

Say that $(q, s) \in \mathcal{L}(A)$ if $A$ has accepting run on $s$ starting in $q$. 
Overview
Intuition of CSHORe
HorSat(T)

Saturation Algorithm for Backwards Reachability

**Input:** A (higher-order) pushdown system $C$ and a finite automaton $A$ recognising a set of configurations $(q, s)$ of $C$.

**Output:** An automaton $\text{pre}^*(A)$ recognising:

$$\mathcal{L}(\text{pre}^*(A)) := \{ (q, s) : (q, s) \longrightarrow^* (q', s') \text{ for some } (q', s') \in \mathcal{L}(A) \}$$

Check whether $(q_0, \bot) \in \mathcal{L}(\text{pre}^*(A))$. 

Broadbent CSHORE: Carayol, Hague, Serre HorsSat: Kobayashi HorSat
Saturation for 1-PDS

Read stack from *top to bottom*.

$q$

$p \xrightarrow{a} b$
Saturation for 1-PDS

Read stack from top to bottom.

(q, b, push\textsubscript{a}, p)

\[(q, [b]) \rightarrow (p, \begin{bmatrix} a \\ b \\ c \end{bmatrix})\]
Saturation for 1-PDS

Read stack from *top to bottom*.

\[(q, b, \text{push}_a, p)\]

\[
(q, [b]) \rightarrow (p, [a])
\]
Read stack from *top to bottom*. 

\[(q, c, \text{pop}, p)\]

\[
\begin{pmatrix}
q \\
\end{pmatrix}
\begin{pmatrix}
c \\
a \\
b \\
\end{pmatrix}
\rightarrow
\begin{pmatrix}
p \\
\end{pmatrix}
\begin{pmatrix}
a \\
b \\
\end{pmatrix}
\]
Saturation for 1-PDS

Read stack from *top to bottom*.

\[
(q, c, \text{pop}, p)
\]

\[
(q, \begin{bmatrix} a \\ b \end{bmatrix}) \rightarrow (p, \begin{bmatrix} a \\ b \end{bmatrix})
\]
Saturation for HORS via CPDS

1. CPDS are a generalisation of pushdown systems enjoying a ‘nested stack structure with pointers’


3. Unlike order-1, naively applying saturation to general $n$-CPDS appears completely impractical.

4. Solution: Combine with an approximate forward analysis
An Optimised Saturation Algorithm

Overview
Intuition of CSHORE
HorSat(T)

Initial

Error
An Optimised Saturation Algorithm
An Optimised Saturation Algorithm
An Optimised Saturation Algorithm
An Optimised Saturation Algorithm
Overview
Intuition of CSHORe
HorSat(T)

An Optimised Saturation Algorithm

Initial
Error
An Optimised Saturation Algorithm
Overview
Intuition of CSHORe
HorSat(T)

Forward Analysis of a Simple 1-PDS

Construct ‘reachability graph’ (e.g. summarization algorithm):

(q_{err}, a)

\[(q_{1}, b)\]

\[(q_{0}, a)\]

\[(q_{0}, \bot)\]

\[(q_{1}, a, \text{push}_{b}, q_{1})\]

\[(q_{1}, a, \text{pop}, q_{0})\]

\[(q_{1}, b, \text{push}_{a}, q_{1})\]

\[(q_{1}, a, \text{pop}, q_{0})\]

\[(q_{0}, b, \text{nop}, q_{0})\]

\[(q_{0}, a, \text{nop}, q_{err})\]

Broadbent CSHORe: Carayol, Hague, Serre HorsSat: Kobayashi
Overview
Intuition of CSHORe

Forward Analysis of a Simple 1-PDS

Extract rules possibly on error trace

(q_{err}, a)

(q_0, \bot, \text{push}_a, q_1) (q_1, a, \text{pop}, q_0)
(q_1, a, \text{push}_b, q_1) (q_1, b, \text{push}_a, q_1)
(q_1, a, \text{push}_a, q_1) (q_1, a, \text{pop}, q_0)
(q_0, a, \text{nop}, q_{err})

(q_0, a)

(q_1, b)

(q_0, \bot, \text{push}_a, q_1)

(q_1, a, \text{push}_b, q_1)

(q_1, a, \text{pop}, q_0)

(q_0, \bot)

(q_1, a)

(q_0, \bot, \text{push}_a, q_1)

(q_1, a, \text{pop}, q_0)

(q_0, b, \text{nop}, q_0)

(q_0, b, \text{nop}, q_0)
Overview
Intuition of CSHORe
HorSat(T)

Forward Analysis of a Simple 1-PDS

Also look at stack symbol after destructive op. . . .

(q₀, ⊥, pushₐ, q₁) (q₁, a, pop, q₀)
(q₁, a, pushₐ, q₁) (q₁, b, pushₐ, q₁)
(q₁, a, pushₐ, q₁) (q₁, a, pop, q₀)
(q₀, a, nop, qerr)

(q₁, b, pushₐ, q₁)
(q₁, b, pushₐ, q₁)
(q₁, a, pop, q₀)

(q₀, a)
(q₁, a)
(q₁, b)
(q₁, a, pushₐ, q₁)
(q₁, a, pushₐ, q₁)
(q₁, a, pop, q₀)
(q₀, b, nop, q₀)

(q₀, ⊥, pushₐ, q₁)
(q₁, a, pop, q₀)

(q₀, ⊥, pushₐ, q₁)
(q₁, a, pop, q₀)

(q₀, a, nop, qerr)
Overview
Intuition of CSHORe
HorSat(T)

Forward Analysis of a Simple 1-PDS

... and ‘guard’ destructive operations

\[(q_0, \bot, push_a, q_1) \quad (q_1, a, pop, q_0)\]
\[(q_1, a, push_b, q_1) \quad (q_1, b, push_a, q_1)\]
\[(q_1, a, push_a, q_1) \quad (q_1, a, pop_\bot, a, q_0)\]
\[(q_0, a, nop, q_{err})\]

\[\quad (q_1, b)\]

\[(q_0, \bot)\]

\[\quad (q_1, a)\]

\[\quad (q_0, a)\]

\[\quad (q_0, a, nop, q_{err})\]

\[\quad (q_1, b, push_a, q_1)\]

\[\quad (q_1, a, pop, q_0)\]

\[\quad (q_0, b, nop, q_0)\]

\[\quad (q_0, a, nop, q_{err})\]

\[\quad (q_0, b)\]

\[\quad (q_0, a, nop, q_{err})\]

\[\quad (q_1, b, push_a, q_1)\]

\[\quad (q_1, a, pop, q_0)\]

\[\quad (q_0, b, nop, q_0)\]

\[\quad (q_0, b)\]
Intuition of CSHORe

Thus have a smaller ‘guarded’ PDS on which to saturate

\[(q_{err}, a)\]

\[
(q_{0}, \perp, push_{a}, q_{1}) \quad (q_{1}, a, pop, q_{0})
\]

\[
(q_{1}, a, push_{b}, q_{1}) \quad (q_{1}, b, push_{a}, q_{1})
\]

\[
(q_{1}, a, push_{a}, q_{1}) \quad (q_{1}, a, pop, q_{0}, q_{err})
\]

\[
(q_{0}, a, nop, q_{err})
\]
Saturation For Guarded Operations

\[(q, c, \text{pop}^\Gamma, p) \quad \text{and} \quad a \in \Gamma\]

\[
\begin{align*}
(q, \begin{bmatrix} c \\ a \\ b \end{bmatrix}) & \rightarrow (p, \begin{bmatrix} a \\ b \end{bmatrix})
\end{align*}
\]
Saturation For Guarded Operations

\[(q, c, \text{pop}^\Gamma, p) \quad \text{and} \quad a \in \Gamma\]

\[
\left( q, \begin{bmatrix} c \\ a \\ b \end{bmatrix} \right) \rightarrow \left( p, \begin{bmatrix} a \\ b \end{bmatrix} \right)
\]
Saturation For Guarded Operations

\[
(q, c, pop^Γ, p) \quad \text{and} \quad a \in Γ
\]

\[
(q, \begin{bmatrix} c \\ a \\ b \end{bmatrix}) \quad \rightarrow \quad (p, \begin{bmatrix} a \\ b \end{bmatrix})
\]
The tool CSHOREe carries out the following work-flow:

1. Compile HORS to CPDS
2. Analyse the CPDS using an ‘approximate summarization’ algorithm (generalised to CPDS)
3. Construct a guarded CPDS
4. Perform Saturation on guarded CPDS

Run-times are usually dominated by step 2—indeed this is often sufficient to establish safety.
Overview
Intuition of CSHORe
HorSat(T)

Higher-Order Model-Checkers

TReCS
Forward Propagation
Concrete Unfolding

TravMC

GTReCS
Game Semantics

HorSat

CSHORE

Backward Propagation
Overview
Intuition of CSHORe
HorSat(T)

Higher-Order Model-Checkers

Intersection Types

TReCS
Forward Propagation
Concrete Unfolding

Game Semantics

GTReCS

TravMC

CSHORE

HorSat

Backward Propagation

Broadbent CSHORE: Carayol, Hague, Serre
HorsSat: Kobayashi
1. Overview

2. Intuition of CSHORE

3. HorSat(T)
Saturation Directly on HORS (HorSat)

1. Far simpler than saturation on CPDS (bordering on trivial)
2. Usually seems to work significantly better in practice
3. Intersection types are ‘standard’ and closer to desirable output
Intersection Types for Model-Checking

Does an automaton $\mathcal{A}$ have a run on tree generated by HORS?

States of $\mathcal{A}$ are atomic types

$$t : q \quad \text{“Unfolding } t \text{ generates tree accepted from } q\text{”}$$

$$f : q' \rightarrow q \quad \text{“feeding } f \text{ a tree accepted from } q'\text{ results in tree accepted from } q\text{”}$$

$$F : (q'' \rightarrow q') \rightarrow q \quad \ldots$$

A transition of the automaton $(p, f, \{(p_1, 1), (p_2, 1), (p_1, 2)\})$ corresponds to a type of terminal $f$:

$$f : (p_1 \land p_2) \rightarrow p_1 \rightarrow p$$
Kobayashi’s Naïve Algorithm

Compute $\Gamma$ assigning types to non-terminals s.t. if

$F : \tau_1 \rightarrow \tau_2 \rightarrow q \in \Gamma$...

...and $F \phi x \rightarrow \phi (G x)$ then

$$\Gamma, \phi : \tau_1, x : \tau_2 \vdash \phi (G x) : q$$

$$\Gamma \vdash \lambda \phi. \lambda x. \phi (G x) : \tau_1 \rightarrow \tau_2 \rightarrow q$$

Greatest fixpoint of $\text{Check}$:

$$\text{Check}(\Gamma) := \{ F : \tau \mid \Gamma \vdash \text{RHS}(F) : \tau \}$$
Kobayashi’s Naïve Algorithm

Compute $\Gamma$ assigning types to non-terminals s.t. if $F : \tau_1 \rightarrow \tau_2 \rightarrow q \in \Gamma$...

...and $F \phi x \rightarrow \phi (G x)$ then

\[
\Gamma, \phi : \tau_1, x : \tau_2 \vdash \phi (G x) : q
\]

\[
\Gamma \vdash \lambda \phi . \lambda x . \phi (G x) : \tau_1 \rightarrow \tau_2 \rightarrow q
\]

Greatest fixpoint of $\mathsf{Check}$:

\[
\mathsf{Check}(\Gamma) := \{ F : \tau \mid \Gamma \vdash \mathsf{RHS}(F) : \tau \}
\]
Kobayashi’s Naïve Algorithm

Compute $\Gamma$ assigning types to non-terminals s.t. if $F : \tau_1 \rightarrow \tau_2 \rightarrow q \in \Gamma$...

...and $F \phi x \rightarrow \phi (G x)$ then

$$\Gamma, \phi : \tau_1, x : \tau_2 \vdash \phi (G x) : q$$

$$\Gamma \vdash \lambda \phi. \lambda x. \phi (G x) : \tau_1 \rightarrow \tau_2 \rightarrow q$$

Greatest fixpoint of $\text{Check}$:

$$\text{Check}(\Gamma) := \{ F : \tau \mid \Gamma \vdash \text{RHS}(F) : \tau \}$$
Saturation Directly on HORS (HorSat)

We consider **infinite** runs accepting (trivial Büchi)

**Greatest** fixpoint of \( \text{Check} \):

\[
\text{Check}(\Gamma) := \left\{ F : \tau \mid \Gamma \vdash \text{RHS}(F) : \tau \right\}
\]

Saturation considers complement of such automata:

Only **finite** runs accepting (trivial co-Büchi)

Types also get dual interpretation: **least** fixpoint of \( \text{Check} \):

\[
\text{Check}(\Gamma) := \left\{ F : \tau \mid \Gamma \vdash \text{RHS}(F) : \tau \right\}
\]
We consider infinite runs accepting (trivial Büchi)

**Greatest** fixpoint of *Check*:

\[
\text{Check}(\Gamma) := \{ F : \tau \mid \Gamma \vdash \text{RHS}(F) : \tau \}
\]

Saturation considers complement of such automata:
Only *finite* runs accepting (trivial co-Büchi)

Types also get dual interpretation: **least** fixpoint of *Check*:

\[
\text{Check}(\Gamma) := \{ F : \tau \mid \Gamma \vdash \text{RHS}(F) : \tau \}
\]
Saturation Directly on HORS (HorSat)

We consider infinite runs accepting (trivial Büchi)

Greatest fixpoint of Check:

\[ \text{Check}(\Gamma) := \{ F : \tau \mid \Gamma \vdash \text{RHS}(F) : \tau \} \]

Saturation considers complement of such automata:
Only finite runs accepting (trivial co-Büchi)

Types also get dual interpretation: least fixpoint of Check:

\[ \text{Check}(\Gamma) := \{ F : \tau \mid \Gamma \vdash \text{RHS}(F) : \tau \} \]
Saturation Directly on HORS (HorSat)

Tree automaton recognising ASTs... the states are types

\[ F \times y \rightarrow G(Hx) \ y \]

\[ @ \]

\[ G \]

\[ H \]

Accept: \( G : p_1 \rightarrow p_2 \rightarrow p \)

\[ H : p_3 \rightarrow p_1 \]
Saturation Directly on HORS (HorSat)

Tree automaton recognising ASTs... the states are types

\[ F \ x \ y \rightarrow G(\text{H}x)\ y \]

\[ @ \]

\[ G \]

\[ H \]

Accept: \[ G : p_1 \rightarrow p_2 \rightarrow p \]

\[ H : p_3 \rightarrow p_1 \]
Tree automaton recognising ASTs... the states are types

\[ F \ x \ y \rightarrow G (Hx) \ y \]

Accept: \[ G : p_1 \rightarrow p_2 \rightarrow p \]
\[ H : p_3 \rightarrow p_1 \]
Overview
Intuition of CSHORe
HorSat(T)

Saturation Directly on HORS (HorSat)

Tree automaton recognising ASTs... the states are types

\[ F(x,y) \rightarrow G(Hx)y \]

Accept:

\[ G: p_1 \rightarrow p_2 \rightarrow p \]
\[ H: p_3 \rightarrow p_1 \]
Saturation Directly on HORS (HorSat)

Tree automaton recognising ASTs... the states are types

\[ F \ x \ y \rightarrow G(Hx)y \]

Accept: \[ G : p_1 \rightarrow p_2 \rightarrow p \quad H : p_3 \rightarrow p_1 \]
Saturation Directly on HORS (HorSat)

Tree automaton recognising ASTs... the states are types

\[ F \ x \ y \rightarrow G(Hx)y \]

\[ \begin{align*}
G & \rightarrow H \\
H & \rightarrow \ldots
\end{align*} \]

Accept: \( G : p_1 \rightarrow p_2 \rightarrow p \) \hspace{1cm} \( H : p_3 \rightarrow p_1 \)
Saturation Directly on HORS (HorSat)

Tree automaton recognising ASTs... the states are types

\[ F \times y \rightarrow G(Hx)y \]

Accept: \( G : p_1 \rightarrow p_2 \rightarrow p \)

\( H : p_3 \rightarrow p_1 \)

\( F : p_3 \rightarrow p_2 \rightarrow p \)
Saturation Directly on HORS (HorSat)

Tree automaton recognising ASTs... the states are types

\[ F \ x \ y \rightarrow G(\text{H}x)\ y \]

\[ \Gamma_N := \{ G : p_1 \rightarrow p_2 \rightarrow p \quad H : p_3 \rightarrow p_1 \} \]
Saturation Directly on HORS (HorSat)

Tree automaton recognising ASTs... the states are types

\[ F \ x \ y \rightarrow G(Hx)y \]

\[ \Gamma_N \vdash G(Hx)y : p \Rightarrow x : p_3, y : p_2 \]

\[ \Gamma_N \vdash G(Hx) : p_2 \rightarrow p \Rightarrow x : p_3 \]

\[ \Gamma_N \vdash y : p_2 \]

\[ \Gamma_N \vdash G : p_1 \rightarrow p_2 \rightarrow p \Rightarrow \emptyset \]

\[ \Gamma_N \vdash Hx : p_1 \Rightarrow x : p_1 \]

\[ \Gamma_N \vdash H : p_3 \rightarrow p_1 \Rightarrow \emptyset \]

\[ \Gamma_N \vdash x : p_3 \Rightarrow x : p_3 \]

\[ \Gamma_N := \{ G : p_1 \rightarrow p_2 \rightarrow p, H : p_3 \rightarrow p_1 \} \]
Saturation Directly on HORS (HorSat)

Tree automaton recognising ASTs... the states are types

\[ F \ x \ y \rightarrow G(Hx)y \]

\[ \Gamma_N \vdash H : p_3 \rightarrow p_1 \Rightarrow \emptyset \]
\[ \Gamma_N \vdash x : p_3 \Rightarrow x : p_3 \]

\[ \Gamma_N \vdash G : p_1 \rightarrow p_2 \rightarrow p \Rightarrow \emptyset \]
\[ \Gamma_N \vdash H x : p_1 \Rightarrow x : p_1 \]

\[ \Gamma_N \vdash G(Hx) : p_2 \rightarrow p \Rightarrow x : p_3 \]
\[ \Gamma_N \vdash y : p_2 \]

\[ \Gamma_N \vdash G(Hx) y : p \Rightarrow x : p_3, y : p_2 \]

\[ \Gamma_N := \{ G : p_1 \rightarrow p_2 \rightarrow p, H : p_3 \rightarrow p_1 \} \]
Naïve HorSat More Formally

Works with complement $\overline{A}$ of property automaton $A$

$$
\Gamma_N \vdash F : \tau \in \Gamma_N
$$

$$
\Gamma_N \vdash F : \tau \Rightarrow \emptyset
$$

$$
\Gamma_N \vdash u : \bigwedge T \rightarrow \tau \Rightarrow \Delta
$$

$$
\Gamma_N \vdash v : \tau' \Rightarrow \Delta' \text{ for each } \tau' \in T
$$

$$
\Gamma_N \vdash (uv) : \tau \Rightarrow \Delta \cup \Delta'
$$

$$(\sim p, f, \{(\sim p_1, 1), (\sim p_2, 2), (\sim p_1, 2)\}) \in \overline{A}
$$

$$
\Gamma_N \vdash f : \sim p_1 \rightarrow (\sim p_2 \land \sim p_1) \rightarrow \sim p
$$

$$
\Gamma_N \vdash \phi : \tau \Rightarrow \phi : \tau
$$

$$
\Gamma'_N = \left\{ F : \tau_1 \rightarrow \cdots \rightarrow \tau_m \rightarrow \sim p \right\}
$$

$$
\Gamma_N \vdash t : \sim p \Rightarrow x_1 : \tau_1, \cdots, x_m : \tau_m
$$
Practical HorSat

Works with complement of property automaton

\[
\frac{F : \tau \in \Gamma_N}{\Gamma_N \vdash F : \tau \Rightarrow \emptyset}
\]

\[
\frac{\Gamma_N \vdash u : \bigwedge T \rightarrow \tau \Rightarrow \Delta \quad \Gamma_N \vdash v : \tau' \Rightarrow \Delta' \text{ for each } \tau' \in T}{\Gamma_N \vdash (uv) : \tau \Rightarrow \Delta \cup \Delta'}
\]

\[
\frac{(\sim p, f, \{(\sim p_1, 1), (\sim p_2, 2), (\sim p_1, 2)\}) \in \overline{A}}{\Gamma_N \vdash f : \sim p_1 \rightarrow (\sim p_2 \land \sim p_1) \rightarrow \sim p}
\]

\[
\frac{\tau \in \text{Inhabit}(\Gamma_N)}{\Gamma_N \vdash t : \tau \Rightarrow \Delta \text{ s.t. } t \text{ might flow to } \phi}
\]

\[
\Gamma'_N = \left\{ \begin{array}{l}
F : \tau_1 \rightarrow \cdots \rightarrow \tau_m \rightarrow \sim p \\
F x_1 \cdots x_m \rightarrow t \text{ and } \\
\Gamma_N \vdash t : \sim p \Rightarrow x_1 : \tau_1, \cdots, x_m : \tau_m
\end{array} \right\}
\]
Whilst HorSat can tell us whether a HORS is safe or not, it cannot provide a certificate for this fact.

In Progress:
Extract certificate from complement of HorSat output (not just complementing a finite automaton—not completely straightforward)

Current solution: HorSatT—run HorSat starting with:

\[ \{ \, F : \top \rightarrow \cdots \rightarrow \top \rightarrow p \mid p \in Q \text{ and } F \text{ N.T.} \, \} \]
Overview
Intuition of CSHORe
HorSat(T)

Higher-Order Model-Checkers

Interception Types

Intersection Types

TReCS
Forward Propagation
Concrete Unfolding
TravMC

Game Semantics

GTReCS

Co-Trivial

HorSat

CSHORE

Backward Propagation

Broadbent CSHORE: Carayol, Hague, Serre HorsSat: Kobayashi
Higher-Order Model-Checkers

- Intersection Types
- TReCS
- GTReCS
- CSHORE
- TravMC
- HorSat
- HorSatT
- Co-Trivial
- Backward Propagation
- Forward Propagation
- Concrete Unfolding
- Game Semantics

Broadbent CSHORE: Carayol, Hague, Serre
HorsSat: Kobayashi
Overview
Intuition of CSHORe
HorSat(T)
Overview
Intuition of CSHORe
HorSat(T)

HorSatT

All Terms
Overview
Intuition of CSHORE
HorSat(T)

HorSatT

All Terms

2

Broadbent CSHORE: Carayol, Hague, Serre
HorsSat: Kobayashi
Overview
Intuition of CSHORe
HorSat(T)
Overview
Intuition of CSHORe
HorSat(T)

HorSatT

All Terms

4 3 2

Broadbent CSHORe: Carayol, Hague, Serre HorsSat: Kobayashi HorSat
Intuition of CSHORe
HorSat(T)

HorSatT

5/6  4  3  2

All Terms

Broadbent CSHORE: Carayol, Hague, Serre HorsSat: Kobayashi
Overview
Intuition of CSHORe
HorSat(T)

HorSatT

5/6

All Terms
Some Experiments

See separate document…
Advantages of Saturation for Higher-Order Model-Checking

- Saturation runs in \textit{linear} time in size of HORS (although CFA0 cubic)
- Working backwards allows one to incrementally construct \textit{accurate} type information
- Offers improved scalability compared to previous tools . . .
- . . . although . . .
A very recent tool [preliminary report at HOPA workshop 2013] (by Steven Ramsay, Robin Neatherway and Luke Ong)
A Possible Way Forward

Develop idea of higher-order saturation further . . .
. . . aim to make higher-order model-checking scalable to more realistic verification problems

Exploit both semantic information given by types and view of type-checking as tree automata [c.f. Bouajjani et al’s abstract regular model-checking]

1 ‘Higher-order predicate abstraction’ for intersections (perhaps eliminate/reduce need for forward analysis... linear run time (under FPT assumptions))
2 Recursive intersection type inference [cf. Kobayashi 2013]
3 Integrating work-flow with existing techniques for refinement-type inference
A Possible Way Forward

Develop idea of higher-order saturation further... 
...aim to make higher-order model-checking scalable to more realistic verification problems

Exploit both semantic information given by types and view of type-checking as tree automata [c.f. Bouajjani et al's abstract regular model-checking]

1. ‘Higher-order predicate abstraction’ for intersections (perhaps eliminate/reduce need for forward analysis... linear run time (under FPT assumptions))
3. Integrating work-flow with existing techniques for refinement-type inference
Overview
Intuition of CSHORe
HorSat(T)

A Possible Way Forward

Develop idea of higher-order saturation further . . .
. . . aim to make higher-order model-checking scalable to more realistic verification problems

Exploit both semantic information given by types and view of type-checking as tree automata [c.f. Bouajjani et al’s abstract regular model-checking]

1. ‘Higher-order predicate abstraction’ for intersections (perhaps eliminate/reduce need for forward analysis... linear run time (under FPT assumptions))
3. Integrating work-flow with existing techniques for refinement-type inference