5. A Model-based Approach to Higher-Order Model Checking

based on "Using Models to Model-Check Recursive Schemes"
Sylvain Salvati & Igor Walukiewicz, LMCS 11(2), 2015

Goal:
- Understand the relationships between
  denotational semantics of the simply-typed λY-calculus
  and
  logical properties of Böhm trees (model-checking problems).
- Give a semantics capturing safety properties.
- Show that semantics defined via extremal fixed points can only capture safety properties.

5.1 λY-Calculus

Goal:
- Introduce the λY-calculus.
- λY-terms generate infinite trees (that may be accepted or rejected by tree automata).
- Tree signatures simplify the presentation.
  Re-blind (insightful (5.2) specific to this study.
  Remaining definitions standard.

Idea: λY-calculus = higher-order recursion schemes with explicit fixed-point operator.

Set of types: τ defined by

$$\alpha ::= o \mid \alpha_1 \to \alpha_2$$

$\to$ associates to the right.

Order of a type:

order(o) := 0

order($\alpha \to \beta$) := max{1 + order(α), order(β)}.

Definition:

- A signature Σ is a set of typed constants, symbols with associated types from $\bar{\tau}$.

- We assume for every type $\alpha \in \bar{\tau}$ there are constants $\underbrace{\omega^\alpha, \Omega^\alpha}_{\text{will stand for undefined terms}}$, $\underbrace{Y^{(\alpha \to \alpha) \to \alpha}}_{\text{will stand for a fixed-point operator.}}$

- Will be interested in two signatures:
  All constants other than ω, Ω, and Y have order at most 1.
  Note that types of order 1 have the shape
  $o^i \to o$, meaning $\underbrace{o \to \dots \to o}_{i\text{-times}} \to o$.
  To simplify the presentation, we will assume all constants in the signatures to have type
  o or $o \to o \to o$.
  This goes okay.

Definition:

The set of simply-typed λ-terms is defined by simultaneous induction over all types:

- Every constant $c \in \Sigma$ of type α is a term of type α.
- For every type α there is a countable set of variables x^α, y^α that are also terms of type α.
- If M is a term of type β and x^α a variable of type α, then $\lambda x^\alpha . M$ is a term of type $\alpha \to \beta$.
- If M is a term of type $\alpha \to \beta$ and N is a term of type α, then MN is a term of type β.

A term is <u>closed</u>, if every variable is bound by a λ-abstraction.

<u>Notation:</u>

1. Application associates to the left.
2. Application binds stronger than λ-abstraction
3. Nested lambdas can be collapsed.

<u>Example:</u>

$$\lambda x y z . x y z$$

(by 2 = $\lambda x y z . (x y z)$

(by 1) = $\lambda x y z . ((x y) z)$

(by 3) = $\lambda x \{\lambda y [\lambda z ((x y) z)]\}$.

Definition:

- The usual operational semantics of the λ-calculus: β-contraction

$$(\lambda x.M)\,N \to_\beta M[N/x].$$

The transition can be applied anywhere in a term.

- To give meaning to fixed-point constants: δ-contraction

$$YM \to_\delta M(YM)$$

Also anywhere in a term.

- We write $\to_{\beta\delta}^*$ for the βδ-reduction.
The reflexive and transitive closure of $\to_\beta$ and $\to_\delta$, $(\to_\beta \cup \to_\delta)^*$.

- Defines an operational equality on terms:
$=_{\beta\delta}$ is the smallest equivalence containing $\to_{\beta\delta}^*$.
Called βδ-conversion or βδ-equality.

- Given $M = \lambda x_1 \dots x_m.\ N_0\, N_1 \dots N_p$
with N_0 of the form

$$(\lambda x.P)\,Q \quad \text{or} \quad YP.$$

Then N_0 is called head redex (reducible expression) of M.

With $M \to_h M'$, if M' obtained by βδ-contracting the head redex of M (if it has one).

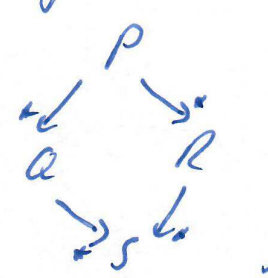
With $\to_h^*$ and $\to_h^+$ for the reflexive transitive closure and the transitive closure of $\to_h$.

Call $\to_h^*$ head reduction.

If term without head redex it is head normal form.

Comments:

- The operational semantics of λY-calculus is $\beta\delta$-reduction.
- This is confluent:



Richard Statman "On the λY-calculus" Annals of Pure and Applied Logic, 2004.

- This enjoys subject reduction:
  The type of terms is invariant under $\beta\delta$-reduction.
- So every term has at most one ($\beta\delta$) normal form
  (where no more β- or δ-contraction can be applied).
- Due to δ-reduction, a term may not have a normal form at all.
- If a term has a head normal form, often called solvable.
  Otherwise, call it unsolvable.
- Even if a term is solvable, it may contain unsolvable subterms and thus does not need to have a ($\beta\delta$) normal form.
  It may even be the case that all subterms are solvable, but reduction generates an infinitely growing term.

$\Rightarrow$ Classical solution in λ-calculus:
Consider an infinite normal form,
an infinite tree that is not a term of λY-calculus.

Böhm trees (Barendregt '84, Amadio & Curien '98)

- Reflects call-by-name
- Call by value can be encoded
- Finite data domains and conditionals can also be encoded.

Idea:

- Unranked, ordered, potentially infinite tree
- Nodes labeled by terms of the form

$$\lambda \vec{x}.\, N$$

where
- N is a variable or a constant $\neq \Omega$.
- the sequence of λ-abstractions may be empty.

Examples:

- x^σ, Ω^σ, $\lambda x^\sigma.\, \omega^\sigma$ are labels
- $\lambda y^\sigma.\, x^{\sigma\to\sigma}\, y^\sigma$ is not.

Definition:

The Böhm tree of a term M is obtained as follows:

- If $M \to_{\beta\delta}^* \lambda \vec{x}.\, N_0\, N_1 \ldots N_k$ with N_0 a variable or a constant $\neq \Omega$,

  then $BT(M)$ has root labeled $\lambda \vec{x}.\, N_0$

  and subtrees $BT(N_1), \ldots, BT(N_k)$.
- Otherwise, $BT(M) = \Omega^\alpha$, where α is the type of M.

Examples:

(1) Consider $Y\ (\lambda F.N)\ a$

with

$$N = \lambda g.\, g\,(b\,(F\,(\lambda x.\, g\,(g\,x))))$$

Here, $a, b : o \to o$

$F, N : \underbrace{(o \to o) \to o}_{\alpha}$

With this,

$$\lambda F.N : \alpha \to \alpha$$

$$Y\,(\lambda F.N) : \alpha$$

Hence, $Y\,(\lambda F.N)\,a : o$

To determine the root of $BT(Y\,(\lambda F.N)\,a)$,
the term $Y\,(\lambda F.N)\,a$ is not of the required form
(variable or constant $\neq Y$).

We have:

$$Y\,(\lambda F.N)\,a$$

$$\to_\delta (\lambda F.N)\,(Y\,(\lambda F.N))\,a$$

$$\to_\beta (N[Y\,(\lambda F.N)/F])\,a$$

$$= (\lambda g.\, g\,[b\,[(Y\,(\lambda F.N))\,(\lambda x.\, g\,(g\,x))]])\,a$$

$$\to_\beta \underbrace{a}_{\text{Constant.}}\,[b\,[(Y\,(\lambda F.N))\,(\lambda x\, a\,(a\,x))]]$$

Hence: $BT(Y\,(\lambda F.N)\,a) = BT(\overset{a}{\overset{|}{}}(b\,(\ldots))) = \overset{a}{\overset{|}{b}}\ldots$

In the end we will have $BT(Y\,(\lambda F.N)\,a) = a\,b\,a^2\,b\,a^4\,b\,a^8 \ldots a^{2^n}\,b \ldots$

(2) Consider

$$Factorial(x) \equiv \underline{if}\ x=0\ \underline{then}\ 1\ \underline{else}\ x.\ Factorial(x-1).$$

We model this as

$$Fct \equiv Y\,[\underbrace{\lambda F.\ \lambda x.\ if\ (zx)\ 1\ (m\ x\ (F\ (-\ x\ 1)))}_{N}]$$

Note that $F : \sigma \to \sigma$

$- : \sigma \to \sigma \to \sigma$.

$z : \sigma \to \sigma$

$if : \sigma \to \sigma \to \sigma \to \sigma$

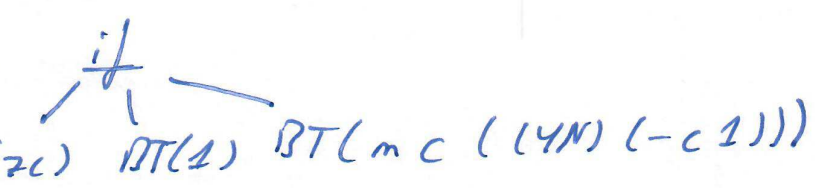
$N : \underbrace{(\sigma \to \sigma)}_{\alpha} \to \underbrace{\sigma \to \sigma}_{\alpha}$

$Y\,N : \alpha$

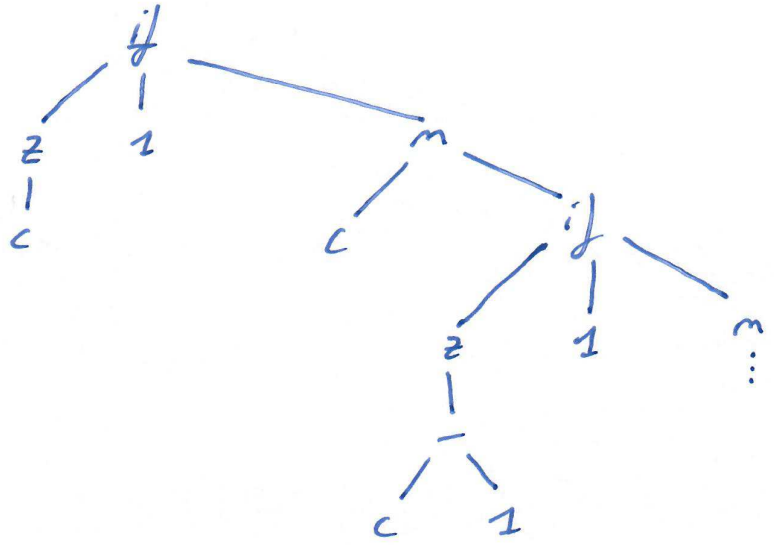
Our goal is to determine $BT(Fct\ c)$.

Its $Fct\ c = Y\,N\,c$ is not of the required form,

$$\begin{aligned}
&Y\,N\,c \\
\to_\delta\ &N\ (Y\,N)\ c \\
\to_\beta\ &[\lambda x.\ if\ (z\,x)\ 1\ (m\ x\ (F\ (-\ x\ 1)))]\,[Y\,N/F]\,c \\
=\ &[\lambda x.\ if\ (z\,x)\ 1\ (m\ x\ ((Y\,N)\ (-\ x\ 1)))]\,c \\
\to_\beta\ &\underbrace{if}_{\text{constant.}}\ (z\,c)\ 1\ (m\ c\ ((Y\,N)\ (-\ c\ 1)))
\end{aligned}$$

Hence, $BT(Fct\ c) =$ 

In the end, the Böhm tree will be



Lemma:
A term M without constants $\mathcal{R}$ and ω
has a $\beta\eta$-normal form iff $BT(M)$ is a finite tree without $\mathcal{R}$.
In this case, $BT(M)$ is just another representation of the normal form.

- Unlike in the standard theory of simply-typed λ-calculus,
we will mostly be interested in terms with infinite Böhm trees.
- Recall that in the signatures, all terms except Y, $\mathcal{R}$, and ω
have type o or $o^2 \to o$.
A closed term without λ-abstraction and Y over such a signature
is just a finite binary tree
  - ↳ Constants of type o form the leaves
  - ↳ Constants of type $o^2 \to o$ form the inner nodes.

The same is true for Böhm trees in the following sense.

Lemma:
If M is a closed term of type o over a tree signature,
then $BT(M)$ is a potentially infinite binary tree.

5.3 Denotational Semantics

Goal: Study finite models of the λY-calculus.
Concentrate on those where Y is interpreted as greatest fixed point
Models interpreting Y as least fixed point:
Dual, capture same class of properties.

Definition:
A GFP-model of a signature Σ
is a tuple $S = ((S_\alpha)_{\alpha \in \mathcal{T}}, \rho)$, where

- S_o is a finite lattice
- $S_{\alpha \to \beta} = mon[S_\alpha \to S_\beta]$ is
the set of monotone functions from S_α to S_β,
ordered pointwise.

The valuation function ρ is required
to satisfy the following conditions:

- If $c \in \Sigma$ is of type α, then $\rho(c) \in S_\alpha$.
- For every $\alpha \in \mathcal{T}$, $\rho(\omega^\alpha)$ and $\rho(\Omega^\alpha)$ are
the greatest element of S_α.
- $\rho(Y^{(\alpha \to \alpha) \to \alpha})$ is the function
assigning to each function $f \in S_{\alpha \to \alpha}$
its greatest fixed point.

Nok:

Since S_α is a finite lattice, it is a complete lattice.
Hence, all GFPs exist without further assumptions (Knaster & Tarski).

Definition:

- A variable assignment is a function ν associating to variables of type α elements of S_α.
If $s \in S_\alpha$ and x^α is a variable of type α,
then $\nu[s/x^\alpha]$ is the variable assignment mapping x^α to s and being identical to ν elsewhere.

- The interpretation of a term M of type α in the model S under assignment ν is an element of S_α, denoted $[\![M]\!]_S^\nu$.
It is defined inductively by

$$[\![c]\!]_S^\nu := S(c)$$

$$[\![x]\!]_S^\nu := \nu(x^\alpha)$$

$$[\![MN]\!]_S^\nu := [\![M]\!]_S^\nu ([\![N]\!]_S^\nu)$$

$$[\![\lambda x^\alpha . M]\!]_S^\nu := \text{a function mapping } s \in S_\alpha \text{ to } [\![M]\!]_S^{\nu[s/x^\alpha]}.$$
We also write this as $\lambda s . [\![M]\!]_S^{\nu[s/x^\alpha]}$.

Lemma: - $[\![M]\!]_S^\nu$ is always monotone.
- $M =_{\beta\delta} N$, then $[\![M]\!]_S^\nu = [\![N]\!]_S^\nu$.
Called: GFP-models are sound wrt. $\alpha\beta$-conversion.

We need a stronger property than soundness:

Same Böhm trees yield the same models.

To define the semantics of a Böhm tree,
need the notion of truncations.

Definition:

- For every $n \in \mathbb{N}$, denote by $BT(M)\downarrow_n$ the finite term obtained by replacing in the tree $BT(M)$ every subtree at depth n by constant ω^α (of appropriate type).
- Since we work with a tree signature:
  If M is closed and of type o,
  then α in ω^α will be o.
- Define
$$[\![BT(M)]\!]^{\vee}_{\mathcal{S}} := \bigcap \{ [\![BT(M)\downarrow_n]\!]^{\vee}_{\mathcal{S}} \mid n \in \mathbb{N} \}$$

Proposition:

If $\mathcal{S}$ is a (finite) GFP-model and M is closed,
then $[\![M]\!]_{\mathcal{S}} = [\![BT(M)]\!]_{\mathcal{S}}$.

Note that in GFP-models, ω^α and $\mathcal{B}^\alpha$ have the same meaning.
Since they are used for different purposes, we use different symbols.

5.4 TAC Automata

Fix a tree signature Σ.
By our assumption, except $\mathcal{B}$, ω, and Y
all constants have type o or $o \to o \to o$.

We only consider closed terms of type o.
By the lemma above, the associated Böhm trees
are potentially infinite binary trees.

Let Σ_0 = constants of type o

Σ_2 = constants of type $o \to o \to o$.

Let $\Sigma = \Sigma_0 \cup \Sigma_2$.

Definition:
A finite tree automaton with trivial acceptance condition (TTAC automaton)
over Σ is a tuple $\mathcal{A} = (Q, \Sigma, q^0, \delta)$
with
- Q a finite set of states, $q^0 \in Q$ the initial state,
- $\delta = (\delta_0, \delta_2)$ the transition function with

$$\delta_0 : Q \times (\Sigma_0 \cup \{\Omega\}) \to \{ff, tt\}$$

$$\delta_2 : Q \times \Sigma_2 \to \mathcal{P}(Q \times Q).$$

A TTAC automaton is Ω-blind, if

$$\delta_0(q, \Omega) = tt \quad \text{for all } q \in Q.$$

Otherwise, the automaton is called insightful.

TTAC automata define languages of possibly infinite binary trees.
Such trees are partial functions

$$t : \{1,2\}^* \to \Sigma \cup \{\Omega\}$$

such that the domain is a binary tree:

(1) If $uv \in dom(t)$ then $u \in dom(t)$

(2) If $u \in dom(t)$ and $t(u) \in \Sigma_2$, then $u.1, u.2 \in dom(t)$.

(3) If $u \in dom(t)$ and $t(u) \in \Sigma_0 \cup \{\perp\}$,
then u is called a leaf
and if $uv \in dom(t)$, then $v = \varepsilon$.

A run of A on t is a mapping

$$r : \{1,2\}^* \longrightarrow Q$$

with
- $dom(r) = dom(t)$ and such that
- $r(\varepsilon) = q^0$ // ε is the root of t
- $(r(u.1), r(u.2)) \in \delta_2(r(u), t(u))$ // u is an inner node.

A run is accepting, if $\delta_0(r(u), t(u)) = tt$ for every leaf u of t.
The tree is accepted by A, if there is an accepting run on it.
The language of A is the set of all trees accepted by A.

Comment:
TAC automata have acceptance conditions only on leaves.
There are no acceptance conditions on infinite paths.
So every run on an infinite tree without leaves is accepting.
This does not imply that TAC automata accept all such trees.
↳ There may actually be no run on a tree.
imagine $\delta_2(q, c) = \emptyset$.

Examples
... of properties that can be expressed with insightful TAC automata

but not with R-blind TAC automata.

(1) The set of terms not having R in their Böhm tree.

Take an automaton with state q.

It has transitions on all letters from Σ_2.

We have

$$\delta_0(q, R) = ff \quad \text{and}$$

$$\delta_0(q, c) = tt \quad \text{for all } c \in \Sigma_0.$$

(2) The set of terms having a head normal form.

Take an automaton with two states, q and $q^\top$.

From $q^\top$, it accepts every tree.

From q, there is a transition to $q^\top$ on all letters from Σ_2.

On Σ_0, the automaton behaves as in Example 1.

(3) Based on the examples, one can construct an automaton for "every occurrence of R is preceded by constant err".

None of these languages is recognized by an R-blind automaton.

This is due to the following closure property of languages of R-blind automata.

Lemma:

Let A be an R-blind TAC automaton.

If $t[t']$ with subtree t' $\in L(A)$, then $t[R] \in L(A)$.

The lemma also shows that the above languages cannot be defined as Boolean combinations of R-blind TAC automata.

5.5 GFP Models and R-Blind TAC Automata

Goal: Show that the recognizing power of GFP models coincides with that of

Boolean combinations of R-blind TAC automata.

„⇐" For every automaton, construct a model discriminating the trees accepted by the automaton.

„⇒" Use Boolean combinations of TAC automata to capture the recognizing power of a model.

What does it mean to recognize by a model?

Definition:

Let S be a GFP model with base set S_0.

The language recognized by $F \subseteq S_0$ is

the set of all closed $\lambda\nu$-terms $\{M \mid [\![M]\!]_S \in F\}$.

Notation:

Given a closed term $M : o$,

see $BT(M)$ as a binary tree $t : \{1,2\}^* \rightharpoonup \Sigma$.

- For every node $v \in \mathrm{dom}(t)$, write M_v for the subtree of t rooted at v.
- The tree $BT(M)_{\downarrow h}$ is the prefix $t_{\downarrow h}$ of t with nodes up to depth h.

There are three types of leaves:

cut leaves labeled ω at depth h.

non-converging leaves labeled $\mathcal{A}$
normal leaves labeled by Σ_0.

- Every node $v \in t_k$ corresponds to a subtree of $BT(\mathcal{A})_{t_k}$, denoted by $\mathcal{A}_v^k$.
  Note that $\mathcal{A}_\varepsilon^k = BT(\mathcal{A})_{t_k}$.

Proposition:
For every $\mathcal{A}$-blind TAC automaton A, $L(A)$ is recognized by a GFP model.

Proof:
We define the model S_A.
The base set is $S_0 = \mathcal{P}(Q)$.
This defines S_α for all types α.
It remains to interpret constants other than ω, $\mathcal{A}$, and Y.

- For $c \in \Sigma_0$, we set

$$S(c) := \{ q \in Q \mid \delta_0(q, c) = \top \}.$$

- For $a \in \Sigma_2$, we set

$$S(a) : \mathcal{P}(Q) \to \mathcal{P}(Q) \to \mathcal{P}(Q)$$

$$(S(a))(Q_0, Q_1) := \{ q \in Q \mid \delta_2(q, a) \cap (Q_0 \times Q_1) \neq \emptyset \}$$

As a recognizer, we take the set

$$F_A := \{ Q' \subseteq Q \mid q^0 \in Q' \}.$$

Lemma:

$BT(M) \in L(\mathcal{A})$ iff $[\![M]\!]_{S_{\mathcal{A}}} \in F_{\mathcal{A}}$.

Proof:

"$\Rightarrow$" Take a Σ_Y-tree M with $BT(M) \in L(\mathcal{A})$.

We have to show that $q^0 \in [\![M]\!]_{S_{\mathcal{A}}}$.

- It will actually be sufficient to show $q^0 \in [\![BT(M)]\!]_{S_{\mathcal{A}}}$, since $[\![BT(M)]\!]_{S_{\mathcal{A}}} = [\![M]\!]_{S_{\mathcal{A}}}$ by the lemma above.

Recall that $[\![BT(M)]\!]_{S_{\mathcal{A}}} = \bigcap \{ [\![BT(M)\downarrow_h]\!]_{S_{\mathcal{A}}} \mid h \in \mathbb{N} \}$

Hence, it is enough to show

$$q^0 \in [\![BT(M)\downarrow_h]\!]_{S_{\mathcal{A}}} \quad \text{for all } h.$$

- Assume we have an accepting run r of $\mathcal{A}$ on $BT(M)$.

By induction on the height of v

in the domain of $BT(M)\downarrow_h$

we show that $r(v) \in [\![M_v^h]\!]_{S_{\mathcal{A}}}$.

The desired inclusion follows with $v = \varepsilon$.

Base case:

- If v is a cut-leaf, then $M_v^h = w^0$.

  So $r(v) \in Q = [\![w^0]\!]_{S_{\mathcal{A}}} = [\![M_v^h]\!]_{S_{\mathcal{A}}}$.

- If v is a non-converging leaf, same argument.

- If v is a normal leaf, then $M_v^h = c \in \Sigma_0$.

  We have

$$r(v) \in \{ q \in Q \mid \delta_0(q, c) = H \} = S(c) = [\![c]\!]_{S_{\mathcal{A}}} = [\![M_v^h]\!]_{S_{\mathcal{A}}}$$

Induction step: If r is an internal node, then

$$M_r^k = a\ M_{r.1}^k\ M_{r.2}^k.$$

By the induction hypothesis,

$r(r.1) \in [\![M_{r.1}^k]\!]_{S_A}$ and

$r(r.2) \in [\![M_{r.2}^k]\!]_{S_A}$.

By definition of $\delta(a)$, we get

$$r(r) \in (\delta(a))([\![M_{r.1}^k]\!]_{S_A}, [\![M_{r.2}^k]\!]_{S_A}) = [\![M_r^k]\!]_{S_A}. \quad \square$$

"$\subseteq$" Take a term M and a state $q \in [\![M]\!]_{S_A}$.

We construct a run of A on $BT(M)$

that starts with q, so $r(\varepsilon) = q$.

- If M has no head normal form, $BT(M) = \Omega$.

The conclusion of a run from q is immediate for any state q,

as the automaton is Ω-blind.

- If M has as head normal form a constant $a \in \Sigma_0$.

The conclusion follows by $[\![a]\!]_{S_A} = \delta(a) = \{q \in Q \mid \delta(q, a) = \top\}$.

- If M has as head normal form

$a\ M_1\ M_2$ with $a \in \Sigma_2$,

then by definition of $[\![a]\!]_{S_A} = \delta(a)$, there are

$(q_1, q_2) \in \delta(a)$ with $q_1 \in [\![M_1]\!]_{S_A}$ and $q_2 \in [\![M_2]\!]_{S_A}$.

We repeat the argument with q_1 from node 1 and q_2 from node 2.

It is immediate to see that the resulting run of A on $BT(M)$

is accepting. ∎

It remains to show that $\mathcal{A}$-blind TAC automata

characterize the power of GFP models:

> Every language recognized by a GFP model
> is a Boolean combination of languages
> of $\mathcal{A}$-blind TAC automata.

In the remainder, fix

a tree signature Σ and

a GFP model $S = ((S_\alpha)_{\alpha \in T}, P)$ over Σ.

We construct a family of automata that reflect S.

Definition:

Let $Q := S_\sigma$, so the set of states
is the base set of the model.

Define

$$\delta_0 : Q \times (\Sigma_0 \cup \{\mathcal{R}(1)\}) \longrightarrow \{\mathit{tt}, \mathit{ff}\}$$

and $\delta_2 : Q \times \Sigma_2 \longrightarrow \mathcal{P}(Q \times Q)$

by $\delta_0(q, a) := \mathit{tt}$ iff $q \leq P(a)$ (order on S_σ)

$$\delta_0(q, a) := \{(q_1, q_2) \mid q \leq (P(a))(q_1, q_2)\}.$$

For $q \in Q$, we define

$$\mathcal{A}_q := (Q, \Sigma, q, \delta).$$

Lemma:

Consider a closed λY-term M of type σ.

Then $BT(M) \in L(\mathcal{A}_q)$ iff $q \leq [\![M]\!]_S$.

Proof:

"⇒" If A_q accepts $BT(M)$, we show $q \in [\![M]\!]_S$.

- Since $[\![M]\!]_S = [\![BT(M)]\!]_S = \bigcap \{ [\![BT(M)\downarrow_k]\!]_S \mid k \in \mathbb{N} \}$,
  we show that for every k we have
  $$q \in [\![BT(M)\downarrow_k]\!]_S .$$
- Fix an accepting run r of A_q on $BT(M)$.
  We show that for every $v \in dom(BT(M)\downarrow_k)$
  we have $r(v) \in [\![M_v^k]\!]_S$.
  This will imply that $r(\varepsilon) = q \in [\![BT(M)\downarrow_k]\!]_S$.
- We proceed by induction on the height of v.

Base case:
- If v is a cut leaf, then $M_v^k = \omega^\sigma$.
  As $[\![\omega^\sigma]\!]_S$ is the greatest element in S_σ,
  we have $r(v) \in [\![M_v^k]\!]_S$.
- For a non-converging leaf, the argument is the same.
- If v is a normal leaf, then $M_v^k = c \in \Sigma_0$.
  Since r is an accepting run, we need to have
  by definition
  $$r(v) \in \delta(c) = [\![M_v^k]\!]_S .$$

Induction step:
- If v is an inner node, then
  $$M_v^k = a\ M_{v.1}^k\ M_{v.2}^k .$$
  By induction, $r(v.i) \in [\![M_{v.i}^k]\!]_S$ for $i = 1, 2$.

Because r is a run, by definition

$$r(v) \in (\delta(a))(r(v.1), r(v.2)).$$

Since $\delta(a)$ is monotone and since

$$r(v.i) \subseteq [\![\mathcal{A}^h_{v.i}]\!]_S,$$

we get

$$r(v) \in (\delta(a))(r(v.1), r(v.2))$$
$$\subseteq (\delta(a))([\![\mathcal{A}^h_{v.1}]\!]_S, [\![\mathcal{A}^h_{v.2}]\!]_S) = [\![\mathcal{A}^h_v]\!]_S.$$

$\subseteq$: Assume $q \in [\![\mathcal{A}]\!]_S$.

- We construct an accepting run of $\mathcal{A}_q$ on $BT(\mathcal{A})$.

  Recall that $\mathcal{A}_v$ is the subtree of $BT(\mathcal{A})$ rooted in node v.

  Take r defined by

  $$r(v) := [\![\mathcal{A}_v]\!]_S.$$

- We show that r is a run of the automaton $\mathcal{A}_{[\![\mathcal{A}]\!]_S}$.

  Since $q \in [\![\mathcal{A}]\!]_S$, this run can easily be turned into a run of $\mathcal{A}_q$ — only change the last transition.

  ↳ By definition, $r(\varepsilon) = [\![BT(\mathcal{A})]\!]_S = [\![\mathcal{A}]\!]_S$.

  ↳ If v is a leaf c, then $r(v) = \delta(c)$ and $\delta_0(\delta(c), c) = \top$.

  ↳ If v is an internal node labelled a,

  then $[\![\mathcal{A}_v]\!]_S = (\delta(a))([\![\mathcal{A}_{v.1}]\!]_S, [\![\mathcal{A}_{v.2}]\!]_S)$,

  so $([\![\mathcal{A}_{v.1}]\!]_S, [\![\mathcal{A}_{v.2}]\!]_S) \in \delta_2([\![\mathcal{A}_v]\!]_S, a)$.

Theorem (Salvati & Walukiewicz 2013):

A language L of λY-terms

is recognized by a GFP model

iff it is a Boolean combination of languages

of $\mathcal{R}$-blind TAC automata.

Proof:

"$\Rightarrow$" Take a model S and $p \in S_o$.

By the above lemma, we get that the language

recognized by p is:

$$L_p = L(\mathcal{A}_p) \setminus \bigcup_{\substack{q \in S_o \\ q \leq p \\ q \neq p}} L(\mathcal{A}_q)$$

This is a Boolean combination of

$\mathcal{R}$-blind TAC automata languages.

So given $F \subseteq S_o$, the language recognized by F

is $\bigcup_{p \in F} L_p$.

"$\Leftarrow$" Take an automaton for every basic language

in the Boolean combination.

Determine the corresponding GFP models

by the above proposition

Make a product of these GFP models.

Take an appropriate F, defined by the form

of the Boolean combination of the basic languages. $\square$