Transfer Theorems

Igor Walukiewicz

Bordeaux University

RECURSION $\equiv$ STACKS

$$F \equiv \lambda x.\ \mathbf{if}\ x = 0\ \mathbf{then}\ 1\ \mathbf{else}\ F(x-1)\cdot x.$$

[Courcelle PhD]:
Recursive schemes $\equiv$ deterministic pushdown automata.

Thm [Senizergues]:
Equivalence of schemes (in terms of trees they generate) is decidable.

Thm [Courcelle]:
MSOL theory of trees generated by schemes is decidable.

What about higher-order schemes?

Second-order scheme

$$Map \equiv \lambda f.\lambda x.\ \mathbf{if}\ x = nil\ \mathbf{then}\ \mathrm{nil}\ \mathbf{else}\ f(hd(x)) \cdot Map(f, tl(x))$$

Thm **[Knapik, Niwiński, Urzyczyn]**:
Higher-order pushdown automata $\equiv$ higher-order safe schemes

Thm **[Parys]**:
Safety is a true restriction

Here:

On decidability of MSO theory of trees generated by higher-order schemes.

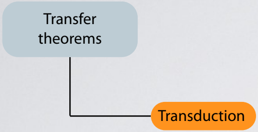
In this talk

Consider an operation $\mathcal{F}$ on models

Transfer property for $\mathcal{F}$
For every φ one can effectively construct $\widehat{\varphi}$, s.t., for every M:

$$\mathcal{F}(M) \vDash \varphi \qquad \text{iff} \qquad M \vDash \widehat{\varphi}.$$

We say in this case that $\mathcal{F}$ is **MSO-compatible.**


Transfer theorems
Transduction

MSO INTERPRETATIONS

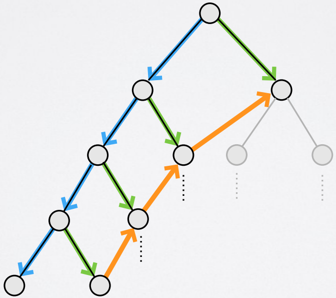
Graph with labelled edges: $G = \langle V, \{E_a\}_{a\in\Sigma}\rangle$

Graph with edge labels from Σ

$\downarrow$

graph with edge labels form Δ

determined by formulas: $\{\varphi_a(x,y)\}_{a\in\Delta}$



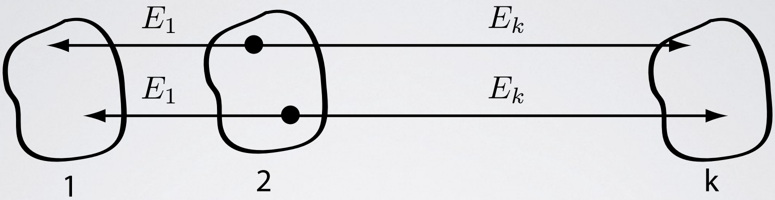
MSO-interpretations are MSO compatible.
For every φ one can effectively construct $\widehat{\varphi}$, s.t., for every M:

$$\mathcal{I}(M) \models \varphi \qquad \text{iff} \qquad M \models \widehat{\varphi}.$$

$$\widehat{\varphi} \equiv \varphi[\varphi_a(x, y) \mapsto E_a(x, y)]$$

k-COPYING

Duplicating k-times a graph $G = \langle V, \{E_a\}_{a\in\Sigma}\rangle$.



$G' = \langle V', \{E'_a\}_{a\in\Sigma}, \{E_i\}_{i\in[k]}\rangle$; where

- $V' = V \times [k]$;
- $E'_a((v, i), (w, i))$ for $(v, w) \in E_a$ and $i \in [k]$;
- $E_i((v, i), (v, j))$ for $v, w \in V$ and $j \in [k]$.

The operation of k-copying is MSO compatible.

MSO-TRANSDUCTIONS

MSO-transduction is a sequence of copying and MSO interpretations

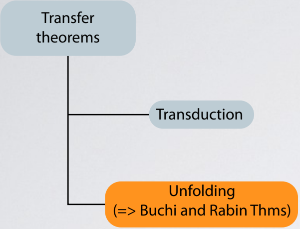
Fact: MSO-transduction is MSO compatible.

$$\begin{array}{ccccccccccc} M_0 & \xrightarrow{copy} & M_1 & \xrightarrow{\mathcal{I}} & M_2 & \ldots & \xrightarrow{copy} & M_{k-1} & \xrightarrow{\mathcal{I}} & M_k \\ \varphi_0 & \longleftarrow & \varphi_1 & \longleftarrow & \varphi_2 & \ldots & \longleftarrow & \varphi_{k-1} & \longleftarrow & \varphi_k \end{array}$$

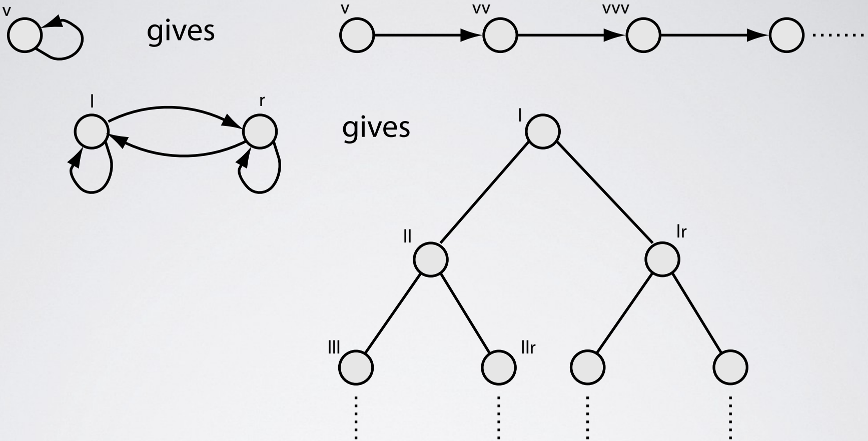
$$M_0 \vDash \varphi_0 \quad \text{iff} \quad M_k \vDash \varphi_k$$

Example: from one node graph we can construct any finite graph.

Remark: Actually it suffices to do one copying and one interpretation.


Transfer theorems
Transduction
Unfolding
(=> Buchi and Rabin Thms)

Unfolding: the tree of all the paths in the graph from a given node.



$Unf(G, v_0) = \langle V^U, \{E_a^*\}_{a \in \Sigma} \rangle$ where

- V^U = paths in G starting from v_0
- $E_a^*(\mathbf{w}v, \mathbf{w}vu)$ if $E_a(v, u)$, and $\mathbf{w} \in V^U$.

Theorem[Courcelle & W.,Muchnik]:
Unfolding is MSO-compatible.
For every $\varphi(x)$ there is (effectively) $\widehat{\varphi}(v_0)$ such that for every graph G and its vertex v_0:

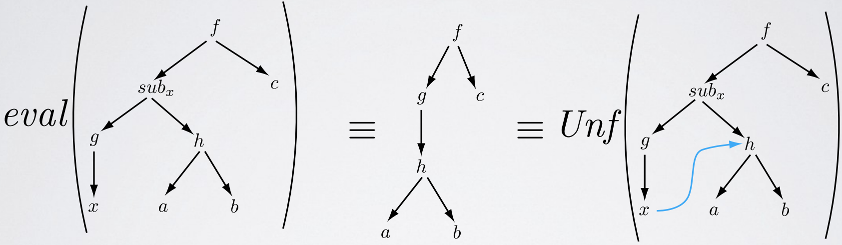
$$G \vDash \widehat{\varphi}(v_0) \qquad \text{iff} \qquad Unf(G) \vDash \varphi(v_0$$

Remark 1: Unfolding cannot be defined by a transduction.

Remark 2: MSO-compatibility of the unfolding implies Büchi and Rabin's Theorems.

Tree with substitutions: function symbols $a, f, g, \dots$; variables $x, y, \dots$; and explicit substitutions sub_x.

$$eval(sub_x(s, t)) = s[t/x]$$

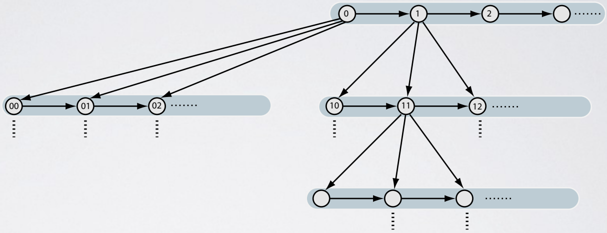


Theorem[Courcelle & Knapik]: For fixed finite set of variables: $eval$ is MSO-compatible

- Transfer theorems
  - Transduction
  - Unfolding (=> Buchi and Rabin Thms)
  - Muchnik Iteration k-tree

Stupp iteration

$St($  $)$ is



$St(G) = \langle V^+, \{E_a^*\}_{a \in \Sigma}, son \rangle$
where for $\mathbf{w} \in V^*$, $u, v \in V$:

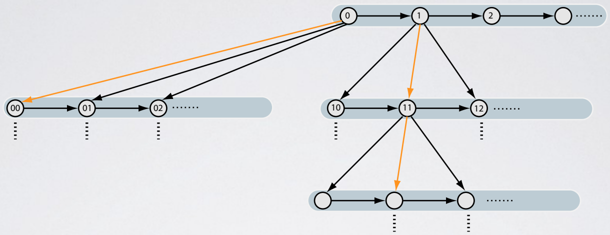
- $son(\mathbf{w}, \mathbf{w}v)$,
- $E_a^*(\mathbf{w}u, \mathbf{w}v)$ when $E_a(u, v)$.

Remark 1: Stupp iteration of the two node graph gives two full binary infinite trees.

Remark 2: Unfolding of a graph may not be definable in the Stupp iteration of the graph.

Remark 3: Stupp iteration of the full binary tree is MSO definable in the full binary tree.

Muchnik iteration

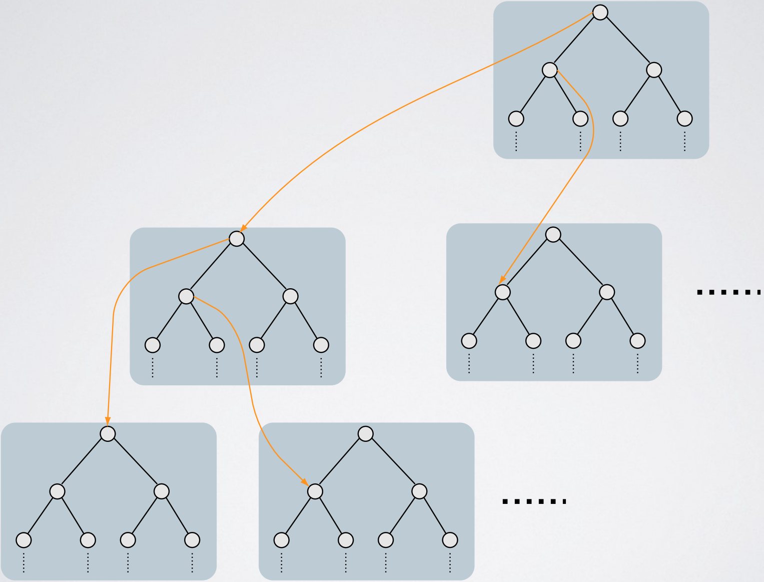


$G^+ = \langle V^+, \{E^*\}_{a\in\Sigma}, E_\#, son\rangle$

- $E_\#(wu, wuu)$ for $w \in V^*$ and $u \in V$.

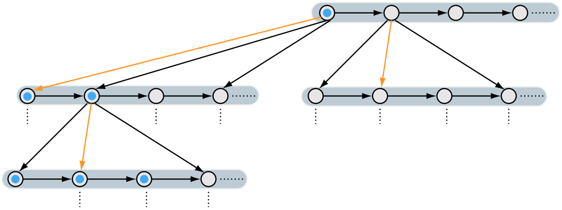
Theorem[Muchnik,W.]: Muchnik iteration is MSO-compatible.

2-tree: Muchnik iteration of the full binary tree.



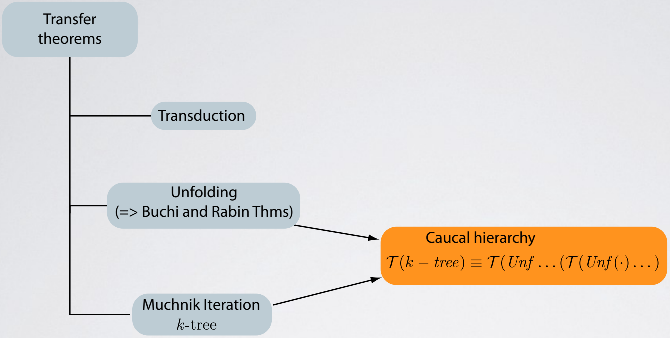
Some things interpretable in k-trees

Interpreting $n(n+1)/2$ in the iteration of the sequence.



Some other things interpretable in k-trees [Fratani & Senizergues]:

- $\langle \mathbb{N}, +1, n\sqrt{n} \rangle$
- $\langle \mathbb{N}, +1, n\log(n) \rangle$
- $\langle \mathbb{N}, +1, n^{k_1},\ n^{k_1 k_2},\ \ldots,\ n^{k_1..k_m} \rangle$


Transfer theorems
Transduction
Unfolding
(=> Buchi and Rabin Thms)
Muchnik Iteration
k-tree
Caucal hierarchy
$\mathcal{T}(k-tree) \equiv \mathcal{T}(Unf \ldots (\mathcal{T}(Unf(\cdot) \ldots)$

Caucal hierarchy

- Level-0: finite graphs
- Level-k: MSO-transductions of k-tree.

Equivalently:

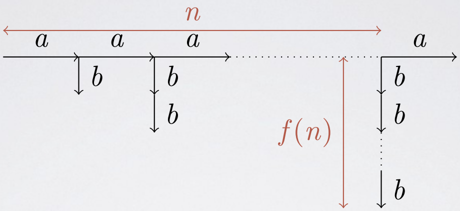
- Level-k: MSO transductions of unfoldings of Level-$(k-1)$ graphs.

$$\mathcal{T}(k - tree) \equiv \overbrace{\mathcal{T}(Unf(\ldots(\mathcal{T}(Unf}^{k}(finite\ graph)\ldots)$$

Cor: All graphs in the Caucal hierarchy have decidable MSO-theory.

Caucal hierarchy is infinite

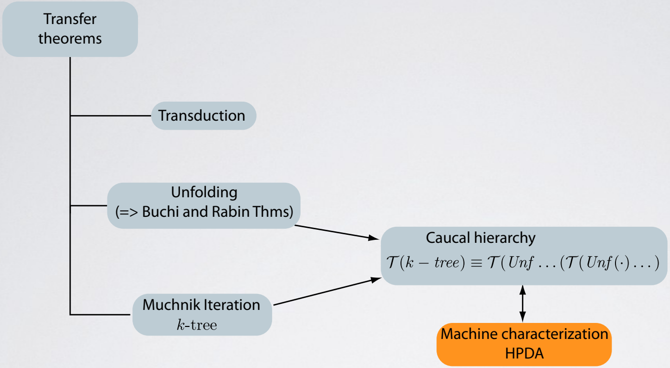
For a function $f : \mathbb{N} \to \mathbb{N}$ we define graph T_f:



Thm[Engelfriet, Carayol & Wöhrle]**:**
$T_{\exp_k}$ graph is a k-level graph but not $(k-1)$-level graph.

Let $\exp_\omega(n) = \exp_n(n)$.

Cor: $T_{\exp_\omega}$ graph is not in the Caucal hierarchy but has decidable MSO theory.


Transfer theorems
Transduction
Unfolding
(=> Buchi and Rabin Thms)
Muchnik Iteration
k-tree
Caucal hierarchy
$\mathcal{T}(k-tree) \equiv \mathcal{T}(Unf \ldots (\mathcal{T}(Unf(\cdot) \ldots)$
Machine characterization
HPDA

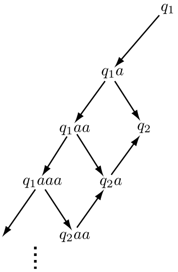
General idea

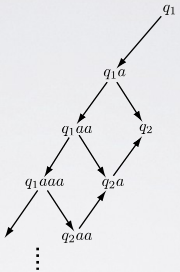
A **graph of configurations** of a machine:

- nodes are configurations of the machine;
- edges represent a step of the computation.

Finite automaton: its graph of configurations is just graph of the automaton

Pushdown automaton:
nodes $qa_1 \dots a_k$
edges
$qa\mathbf{w} \to q\mathbf{w}$ or
$qa\mathbf{w} \to qba\mathbf{w}$.





Configuration graph of a pushdown automaton is interpretable in a tree
Cor: It has decidable MSO-theory

Rem: Turing Machine graphs may have undecidable MSO-theory.

2-nd order stack: example

A 2-**stack** is a stack of stacks. $[a^1_1 \dots a^1_{k_1}][a^2_1 \dots a^2_{k_2}] \dots [a^n_1 \dots a^n_{k_n}]$

New operation of copying the top-most stack:
$q[w_1] \dots [w_i] \to q[w_1][w_1] \dots [w_i]$.

$q_1[a]$	$\to$	$q_1[aa]$	$\to$	$\cdots$	$\to$	$q_1[a^k] \to$
$q_2[a^k]$	$\to$	$q_2[a^{k-1}][a^k]$	$\to$	$\cdots$	$\to$	$q_2[a][aa] \dots [a^k] \to$
$q_3[a][aa] \dots [a^k]$	$\to$	$q_3[aa] \dots [a^k]$	$\to$	$\cdots$	$\to$	$q_3[a^k] \to q_3\bot$

A system where all paths are of the form $q_1^k q_2^k q_3^k$.

Remark: The 2-stack gives additional power.

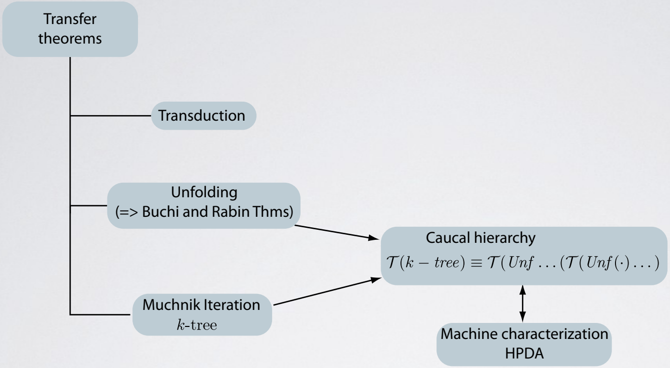
Remark: The above automaton recognizes $\{a^k b^k c^k : k \in \mathbb{N}\}$.

Higher order pushdowns ≡ Caucal hierarchy

- Configuration graph of a pushdown automaton is interpretable in a tree
- Configuration graph of a k-pushdown automaton is interpretable in a k-tree.

Cor: All these graphs have decidable MSO-theory.

Thm[Carayol & Wöhrle]:
Graphs of Caucal level k are configuration graphs of k-th order pushdown automata. (when ε-transitions are contracted).



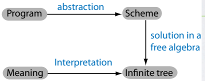
$\lambda Y - calculus$

Schemes

+ Ianov'58 "The logical schemas of algorithms"

+ Park PhD'68 Recursive schemes

+ Scott, Elgot



+ Milner'73 Plotkin'77 PCF

Languages, Higher-order pushdowns

+ Aho'68 indexed languages

+ Maslov'74 '76 higher-order indexed languages and higher order pushdown automata.

+ Courcelle'76 for trees: 1-st order schemes=CFL

+ Engelfriet Schmidt'77 IO/OI

+ Damm'82 for languages: rec schemes= higher-order pusdowns

+ Kanpik Niwinski Urzyczyn'02 Safe schemes = higher-order pusdown

+ Senizergues'97 Equivalence of 1st order schemes is decidable

+Statman'04 Equivalence of PCF terms is undecidable

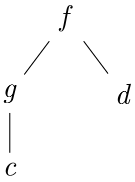
+Loader'01: Lambda-definability is undecidable

Simply typed λ-calculus with fixpoints

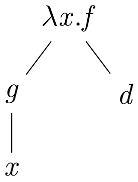
- Types: 0 is a type, and $\alpha \to \beta$ is a type if α, β types.
- Constants: c^α of type α.
- Terms: c^α, x^α, MN, $\lambda x^\alpha.M$.

Example: $c, d : 0$, $g : 0 \to 0$, $f : 0 \to 0 \to 0$

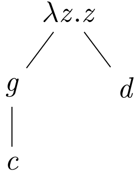
$f(gc)d : 0$



$\lambda x.f(gx)d : 0 \to 0$



$\lambda z.z(gc)d : (0 \to 0 \to 0) \to 0$



β-reduction: $(\lambda x.M)N =_\beta M[N/x]$

$(\lambda x.f(gx)d)c \to_\beta f(gc)d$

$(\lambda z.z(gc)d)(\lambda xy.y) \to_\beta (\lambda xy.y)(gc)d \to_\beta d$

Substitution is as in logic: one should avoid variable capture

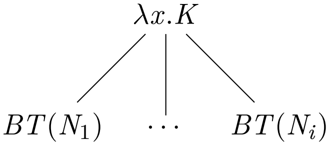
$(\lambda h.\lambda x.g(hx))(fx) \to_\beta \lambda y.g(fxy)$

and not $\lambda x.g(fxx)$

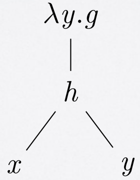
$$f : 0 \to 0 \to 0, \quad g, h : 0 \to 0$$

A **Böhm tree of a term M:**

- We reduce M to head normal form:
  $M \to_\beta^* \lambda \vec{x}.KN_1 \dots N_i$ with K a variable or a constant.
- $BT(M)$ is



Böhm tree of $(\lambda y.g(hxy))$ is



Where are trees?

$$c : 0,\ g : 0 \to 0,\ f : 0 \to 0 \to 0$$

If $M : 0$ is a closed term, and M in head normal form then $M \equiv K N_1 \dots N_i$ with K a constant. So it is either:

$$\begin{tikzcd} & f \arrow[dl, no head] \arrow[dr, no head] & \\ N_1 & & N_2 \end{tikzcd} \quad \text{or} \quad \begin{tikzcd} g \arrow[d, no head] \\ N_1 \end{tikzcd} \quad \text{or} \quad c$$

with $N_0, N_1 : 0$. Hence $BT(M)$ is a ranked tree.

Order of type: *Ord*$(0) = 0$, *Ord*$(\alpha \to \beta) = \max($*Ord*$(\alpha) + 1,$ *Ord*$(\beta))$.

First order signature: all constants of order ≤ 1.

Remark: For closed $M : 0$ over a first order signature $BT(M)$ is a ranked tree.

λY-CALCULUS

We add constants $Y^{(\alpha\to\alpha)\to\alpha}$ and ω^α, for every type α.

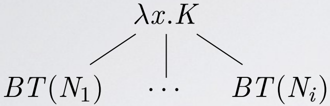
New reduction rule $YM \to_\delta M(YM)$.

Example: YM with $M = (\lambda x.ax)$

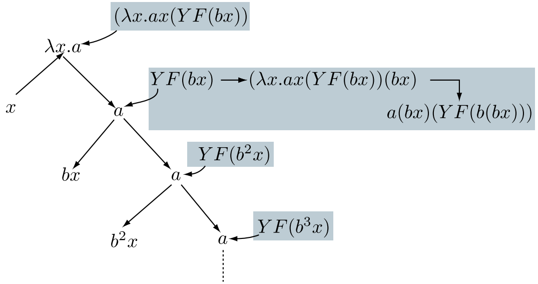
$$\begin{aligned} YM &\to_\delta M(YM) \equiv (\lambda x.ax)(YM) \\ &\to_\beta a(YM) \\ &\to_\delta a(M(YM)) \\ &\to_\beta a(a(YM)) \to \ldots \end{aligned}$$

A **Böhm tree of a λY-term** M is:

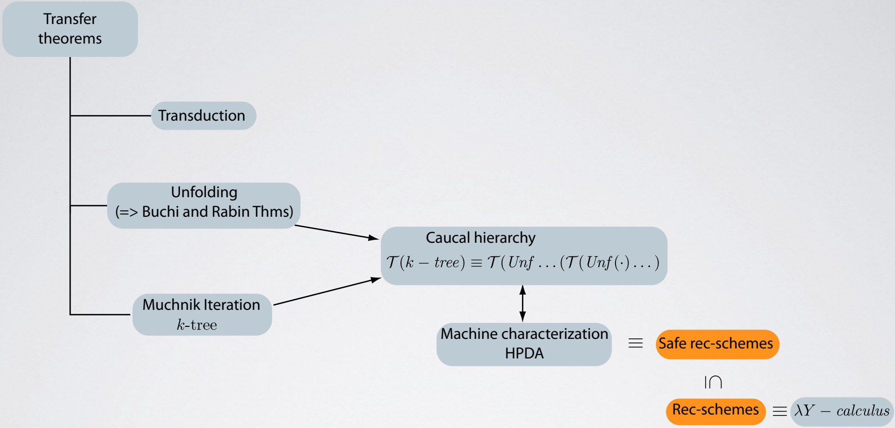
- If M has no head normal form then ω^{α}.
- Otherwise $\lambda\vec{x}.KN_1 \ldots N_i$ is the head normal form and $BT(M)$ is



$Y(\lambda F.\lambda x.ax(F(bx))) \; : \; 0 \to 0$



For closed terms of type 0 over first-order signatures, Böhm tree is a tree.


Transfer theorems
Transduction
Unfolding
(=> Buchi and Rabin Thms)
Muchnik Iteration
k-tree
Caucal hierarchy
$\mathcal{T}(k-tree) \equiv \mathcal{T}(Unf \dots (\mathcal{T}(Unf(\cdot)\dots)$
Machine characterization
HPDA
$\equiv$
Safe rec-schemes
$\subseteq$
Rec-schemes
$\equiv$
$\lambda Y - calculus$

Recursive program schemes

First example

- $F \equiv \lambda x.\ \mathbf{if}\ x = 0\ \mathbf{then}\ 1\ \mathbf{else}\ F(x-1) \cdot x$.
- Abstract form: $F = \lambda x.\ c\ (zx)\ a\ (m\ (F(px))\ x)$.

Another program with the same abstract form:
$Rev \equiv \lambda x.\ \mathbf{if}\ x = \mathrm{nil}\ \mathbf{then}\ \mathrm{nil}\ \mathbf{else}\ Rev(\mathrm{tl}(x)) \cdot hd(x)$

Second-order scheme

$Map \equiv \lambda f.\lambda x.\ \mathbf{if}\ x = nil\ \mathbf{then}\ \mathrm{nil}\ \mathbf{else}\ f(hd(x)) \cdot Map(f, tl(x))$

Order of a scheme: maximal order of a “nonterminal”.

Semantics

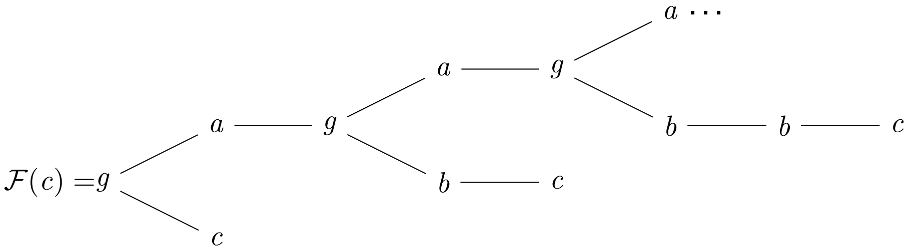
Example:

$$\mathcal{F} = \lambda x.\ a(\mathcal{F}(bx))$$

$$\mathcal{F}(c) \rightarrow_{\beta,\delta} a(\mathcal{F}(bc)) \rightarrow_{\beta,\delta} a(a(\mathcal{F}(b(bc)))) \rightarrow_{\beta,\delta} \ldots$$

Semantics as a tree of execution

$$\mathcal{F} = \lambda x.\ g\ (a(\mathcal{F}(bx)))\ x$$



Recursion schemes $\equiv \lambda Y$-calculus

$$
\begin{aligned}
F_1 &= \lambda \vec{x}. M_1 \\
&\vdots \\
F_n &= \lambda \vec{x}. M_n
\end{aligned}
$$

$$
\begin{aligned}
T_1 &= Y(\lambda F_1. M_1) \\
T_2 &= Y(\lambda F_2. M_2)[T_1/F_1]) \\
&\vdots \\
T_n &= Y(\lambda F_n. (\ldots((M_n[T_1/F_1])[T_2/F_2])\ldots)[T_{n-1}/F_{n-1}])
\end{aligned}
$$

Fact

The tree generated from F_n is $BT(T_n)$.
There is also a translation from λY-terms to schemes.

Theorem[Courcelle]:
The meanings of 1-st order recursive schemes $\equiv$ unfoldings of pushdown graphs.

Theorem[Knapik, Niwiński & Urzyczyn]:
n-th order safe schemes $\equiv$ unfoldings of n-th order pushdown graphs.

Safe $\approx$ no parameters in recursion $\approx$ no problems with static links

Safety

Variables that occur free in a safe λ-term have orders no smaller than that of the term itself.

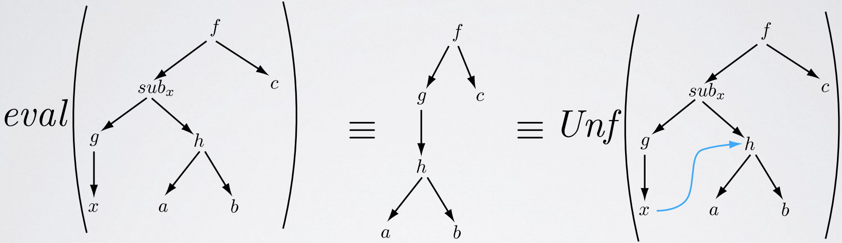
Safe $\Rightarrow$ no need to perform variable renaming when doing β-reduction.

$$(\lambda x^{\alpha}.M)N^{\alpha} \ \rightarrow_{\beta} \ M[N^{\alpha}/x^{\alpha}]$$

$(\lambda y^{\beta}.K)^{\beta\rightarrow\gamma}\,[N^{\alpha}/x^{\alpha}]$ $\qquad$ β has smaller order than α

Tree with substitutions: function symbols $a, f, g, \ldots$; variables $x, y, \ldots$; and explicit substitutions sub_x.

$$eval(sub_x(s,t)) = s[t/x]$$



Theorem[Courcelle & Knapik]: For fixed finite set of variables: $eval$ is MSO-compatible

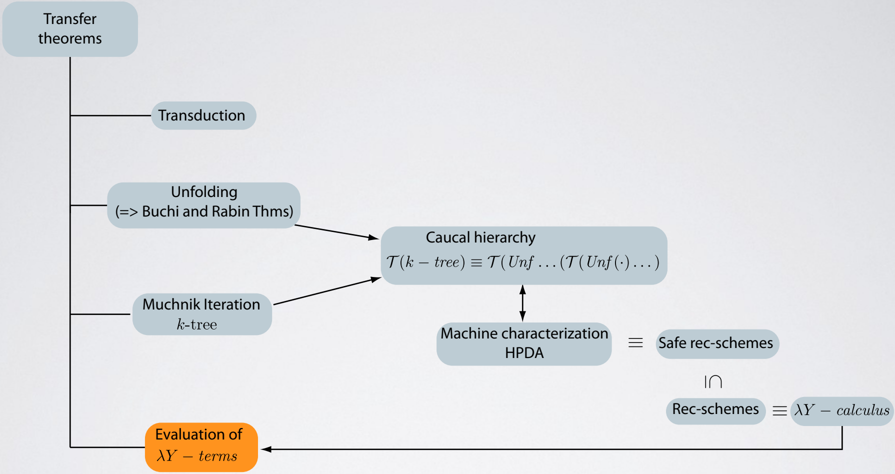
What about schemes that are not safe?

New operation of panic on 2-stack, and then collapse on a higher-order stack. [Urzyczyn, Knapik & Niwiński & Urzyczyn & W., Hague & Murawski & Ong & Serre]

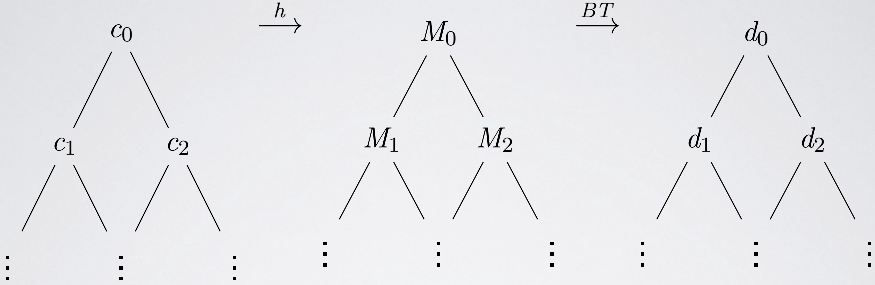
Theorem[Hague & Murawski & Ong & Serre]:
n-th order schemes $\equiv$ unfoldings of n-th order collapse pushdown graphs.

Theorem[Parys]:
Urzyczyn's scheme is not equivalent to a safe scheme.

Theorem[Ong]:
MSO theory of the tree generated by a recursive scheme is decidable.



Transfer theorems
Transduction
Unfolding (=> Buchi and Rabin Thms)
Caucal hierarchy
$\mathcal{T}(k-tree) \equiv \mathcal{T}(Unf \ldots (\mathcal{T}(Unf(\cdot) \ldots)$
Muchnik Iteration k-tree
Machine characterization HPDA
$\equiv$
Safe rec-schemes
$\subseteq$
Rec-schemes
$\equiv$
$\lambda Y - calculus$
Evaluation of $\lambda Y - terms$


c_0
c_1
c_2
h
M_0
M_1
M_2
BT
d_0
d_1
d_2

Signature $\Sigma = (B, C)$

- B - a set of base types
- C - a set of constants with types in $\mathit{Types}(B)$.

Terms over Σ defined as usual.

Homomorphism, for two signatures $\Sigma_1 = (B_1, C_1)$, $\Sigma_2 = (B_2, C_2)$, is a function

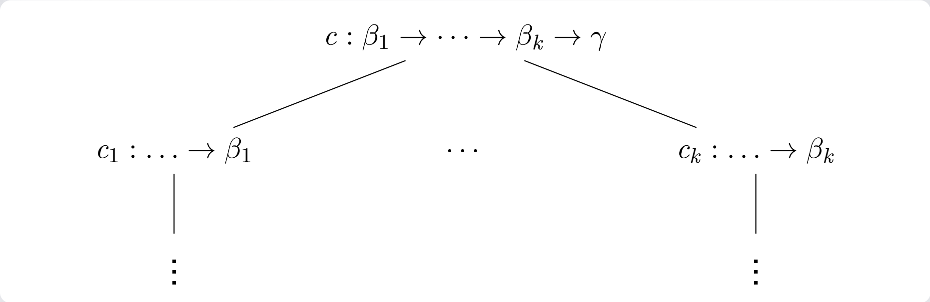
$$h : B_1 \to \mathit{Types}(B_2) \qquad h : C_1 \to \mathit{Terms}(\Sigma_2)$$

with the restriction that $h(c^\alpha)$ is term of type $h(\alpha)$.

First-order signature $\Sigma = (B, C)$:
all constants in C have types of order ≤ 1

$$c : \beta_1 \to \dots \to \beta_k \to \gamma \qquad \text{with} \qquad \beta_1, \dots, \beta_k, \gamma \in B.$$

Applicative tree: well typed term (infinite) of a base type constructed only from constants.



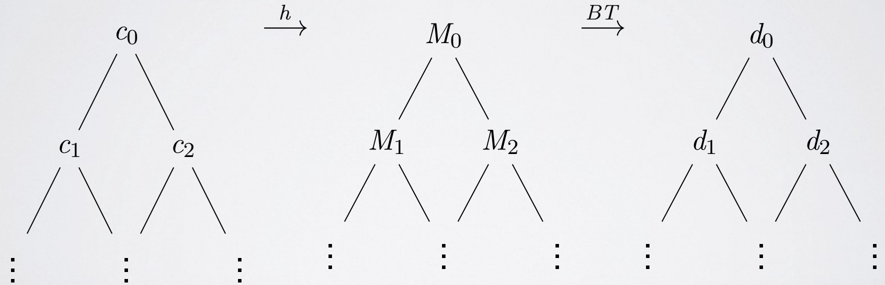
Rem: Applicative trees are just ranked trees so we can talk about their MSO-theories.

First-order signatures $\Sigma_1 = (B_1, C_1)$, $\Sigma_2 = (B_2, C_2)$ and a homomorphism

$$h : B_1 \to \mathit{Types}(B_2) \qquad h : C_1 \to \mathit{Terms}(\Sigma_2)$$

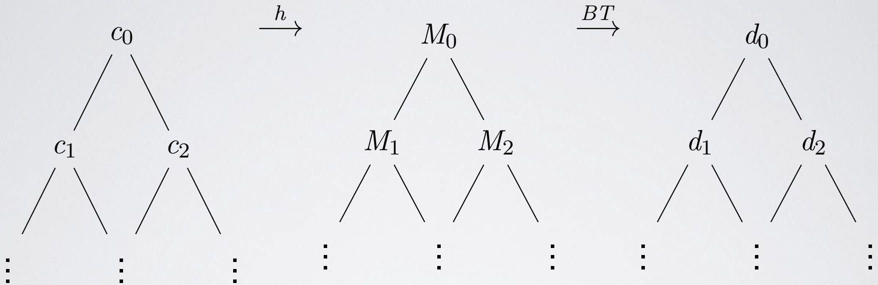
such that $h(\gamma)$ is a base type.

If $t : \gamma$ is an applicative tree over Σ_1 then $BT(h(t))$ is an applicative tree over Σ_2.



Tree operation $t \mapsto BT(h(t))$.

Tree operation $t \mapsto BT(h(t))$



Thm[Salvati & W.]:
Operation $\quad t \mapsto BT(h(t)) \quad$ is MSO compatible.

For every φ there is $\widehat{\varphi}$ s.t. for every applicative tree t of type γ:

$$BT(h(t)) \vDash \varphi \qquad \text{iff} \qquad t \vDash \widehat{\varphi}$$

Take an λY-term M and $c : \gamma$. Set $h(c) = M$. We get:

$$BT(h(c)) \vDash \varphi \qquad \text{iff} \qquad c \vDash \widehat{\varphi}$$

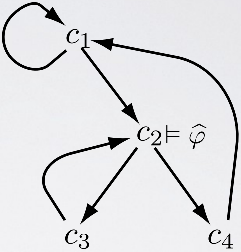
This is Ong's theorem: $BT(M)$ has decidable MSO-theory.

Remark: Every tree in a Caucal hierarchy is $\mathcal{I}(BT(M))$ for some M.

Thm[Parys]: $BT(M)$ may be outside Caucal hierarchy.

Scheme of recursive calls

$$BT(t|_{c_2}) \vDash \varphi$$



Each call represents a procedure $h(c_i) = M_i$.

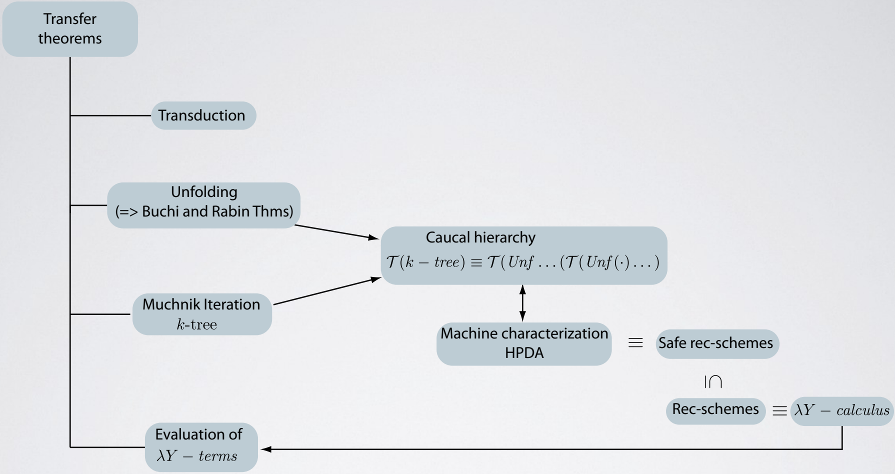
Given a property φ we can say at which recursive calls it holds.

We have modules $M_1, \ldots, M_k$.
Can we write a program with these modules
whose execution satisfies φ?

Take homomorphism $h(c_i) = M_i$:

$$BT(h(t)) \vDash \varphi \qquad \text{iff} \qquad t \vDash \widehat{\varphi}$$

- Any $t \vDash \widehat{\varphi}$ gives a program $h(t)$ satisfying φ.
- If $\widehat{\varphi}$ is satisfiable then there is a regular t
- Using the fixpoint combinator we get a finite program $h(t)$.


Transfer theorems
Transduction
Unfolding
(=> Buchi and Rabin Thms)
Caucal hierarchy
$\mathcal{T}(k-tree) \equiv \mathcal{T}(Unf \ldots (\mathcal{T}(Unf(\cdot)\ldots)$
Muchnik Iteration
k-tree
Machine characterization
HPDA
$\equiv$
Safe rec-schemes
$\subseteq$
Rec-schemes
$\equiv$
$\lambda Y - calculus$
Evaluation of
$\lambda Y - terms$