On Properties of Higher-Order Languages

Naoki Kobayashi
The University of Tokyo

Our Recent Research

(joint work with C. Broadbent, A. Igarashi, K. Matsuda, R. Sato, A. Shinohara, T. Terauchi, T. Tsukada, H. Unno, ...)

- Theory and Practice of Higher-Order Model Checking (model = higher-order grammar)
  - properties of higher-order grammars [LICS13]
  - higher-order model checking algorithms [JACM 13, FoSSaCS11,CSL13, ESOP13, APLAS13]
  - automated verification of higher-order programs [JACM 13, PLDI11,FLOPS12,PEPM13,POPL13]
  - data compression (generalization of grammar-based approach) [PEPM12]

Our Recent Research

- Theory and Practice of Higher-Order Model Checking (model = higher-order grammar)
  - properties of higher-order grammars [LICS13]
  - higher-order model checking algorithms [JACM 13, FoSSaCS11,CSL13, ESOP13, APLAS13]
  - automated verification of higher-order programs [JACM 13, PLDI11,FLOPS12,PEPM13,POPL13]
  - data compression (generalization of grammar-based approach) [PEPM12]

This Talk

- A survey on higher-order (formal) languages
  - what are higher-order languages?
  - solved/open problems
- Applications of λ-calculus and types to studies of higher-order languages
  - Pumping lemma for higher-order recursion schemes (HORS) [K, LICS13]
  - Context-sensitiveness (ongoing work)

Outline

- Background
  - What are higher-order languages and what are they for?
  - Some variations
    - deterministic/non-deterministic, safe/unsafe, OI/IO, word/tree
- Solved/open problems
  - decision problems
  - language hierarchy
- Applications of typed λ-calculus

Higher-Order Grammars
[Maslov74,Wand75,...]

- Extension of CFG where non-terminals take parameters (cf. macro grammar)

Example of order-1 grammar G_1

$S \to A\ e$

$A\ x \to x$

$A\ x \to a\ (A\ (b\ x))$

$S: o,\ A: o \to o$

$S \to A\ e \to a(A\ (b\ e)) \to a(b\ e)$

$L(G_1) = \{a^n b^n e \mid n \geq 0\}$

Higher-Order Grammars

- Extension of CFG where non-terminals take parameters

Example of order-2 grammar G_2	
$S \to A\ b$	$T\ f\ x \to f\ (f\ x)$
$A\ f \to f\ e$	$A\ f \to a\ (A\ (T\ f))$
S: o, A: (o→o)→o, T: (o→o)→o→o	

$$S \to A\ b \to a(A\ (T\ b)) \to^* a^n(A(T^n\ b)))$$

$$\to a^n(T^n\ b\ e) \quad \to^* a^n(b^{2^n} e)$$

$$L(G_2) = \{a^n b^{2^n} e \mid n \geq 0\}$$

Why Higher-Order Languages?

- **Semantics of programs**
  ("recursive program schemes" [Park68,Nivat72,...]**)**
- **Natural extension of Chomsky hierarchy**
  [Wand74,Damm82,..] **(order-0 = regular,**
  order-1 = context-free, order-2 = indexed)
- **Verification of higher-order programs**
  (higher-order grammars as natural models of
  functional programs [Knapik+02, Ong06, K09,...]**)**
  - **generalization of model checking approach to program verification (order-0 = finite state m.c., order-1 = pushdown m.c.)**

Classification of model checking

	model	corresponding inclusion problem (*)	software model checkers
finite state m.c.	automata, regular languages	regular $\subseteq$ regular	BLAST (for C)
pushdown m.c.	pushdown, context-free grammars	context-free $\subseteq$ regular	SLAM (for C)
higher-order m.c. [Knapik+02] [Ong06]	higher-order pushdown, higher-order grammars	higher-order $\subseteq$ regular	MoCHi (for ML) [K+ PLDI11]

(*) infinite words/trees may be considered

Outline

- Background
  - What are higher-order languages and for what?
  - Some variations
    - deterministic/non-deterministic, safe/unsafe
- Solved/open problems
  - decision problems
  - language hierarchy
- Applications of typed λ-calculus

Deterministic vs Non-Deterministic Grammars

- **Deterministic** (aka higher-order recursion schemes):
  - **exactly one rule** for each non-terminal
  - generates a **single** (possibly infinite) tree
    (word, if the arities of terminal symbols are at most 1)
  - models of higher-order model checking
    [Knapik+ 02,Ong06]
- **Non-deterministic:**
  - **an arbitrary number of rules** for each non-terminal
  - generates a **language of** (usually finite) trees
    (words, if the arities of terminal symbols are at most 1)
  - further classification based on evaluation order
    (OI and IO [Damm82])

Safe vs Unsafe Grammar

- safe (“derived types” [Damm82])
  - The arguments of non-terminals are sorted in the decreasing order of their type-theoretic orders
    - ✓ $((o\to o)\to o) \to (o\to o) \to (o\to o\to o) \to o \to o$

      order-2 order-1 order-0

order(o) = 0
order($\tau_1 \to \tau_2$)=max(order(τ_1)+1, order(τ_2))

Safe vs Unsafe Grammar

- safe ("derived types" [Damm82])
  - The arguments of non-terminals are sorted in the decreasing order of their type-theoretic orders

✓ $\underbrace{((o \to o) \to o)}_{\text{order-2}} \to \underbrace{(o \to o)}_{\text{order-1}} \to \underbrace{(o \to o \to o)}_{\text{order-1}} \to \underbrace{o}_{\text{order-0}} \to o$

✗ $\underbrace{(o \to o)}_{\text{order-1}} \to \underbrace{o}_{\text{order-0}} \to \underbrace{(o \to o)}_{\text{order-1}} \to o$

$$\mathrm{order}(o) = 0$$
$$\mathrm{order}(\tau_1 \to \tau_2) = \max(\mathrm{order}(\tau_1) + 1, \mathrm{order}(\tau_2))$$

Safe vs Unsafe Grammar

- safe
  - The arguments of non-terminals are sorted in the decreasing order of type-theoretic orders
    - ✓ $((o\to o)\to o) \to (o\to o) \to (o\to o\to o) \to o \to o$
    - ✗ $(o \to o) \to o \to (o \to o) \to o$
  - Arguments of the same order must be passed at the same time

Non-example:
S -> F h a a.   F z x y -> f (F (F z y) y (z x)) x.
(S:o, F: $(o\to o)\to o\to o \to o$, f:$o\to o\to o$, h:$o\to o$, a:o)

Safe vs Unsafe Grammar

- safe
  - The arguments of non-terminals are sorted in the decreasing order of type-theoretic orders
    - ✓ $((o \to o) \to o) \to (o \to o) \to (o \to o \to o) \to o \to o$
    - ✗ $(o \to o) \to o \to (o \to o) \to o$
  - Arguments of the same order must be passed at the same time
  - Targets of earlier studies [Damm82, Knapik+02,...]
  - Equivalent to higher-order pushdown automata [Maslov74,Knapik+02,...]
- unsafe
  - Without safety restriction
  - Targets of recent studies [Ong06,K09,...]
  - Equivalent to collapsible pushdown automata [Hague+08]

Outline

- Background
  - What are higher-order languages and for what?
  - Some variations
- Solved/open problems
  - decision problems
  - language hierarchy
- Applications of typed λ-calculus

Some decision problems

- **Model Checking**
  - Input: deterministic HO grammar (HORS) G
    regular (or MSO-definable) language R
    (of infinite trees)
  - Output: Tree(G) $\in$ R?
- **Language inclusion**
  - Input: non-deterministic HO grammar G,
    regular language R (of finite words/trees)
  - Output: L(G)$\subseteq$ R?
- **Equivalence**
  - Input: deterministic HO grammars (HORS) G_1, G_2
  - Output: Tree(G_1) = Tree(G_2)?

Problem Status

	safe	unsafe
model checking	decidable [Knapik+02]	decidable [Ong06]
inclusion	decidable [Damm82?]	decidable
equivalence	open (for order ≥ 2)	open (for order ≥ 2)

Language hierarchies

- Language/tree classes
  - $LANG_n$ = {L(G) | G: order-n n.d. word grammar}
  - $TREE_n$ = {Tree(G) | G: order-n HORS}
- Some questions about language hierarchy
  - strictness of word language hierarchy
    $LANG_n \neq LANG_{n+1}$ for all n?
  - strictness of tree hierarchy
    $TREE_n \neq TREE_{n+1}$ for all n?
  - context-sensitiveness
    Is L(G) context-sensitive for all G?
  - Is safety a genuine restriction?
    $SafeLANG_n = UnsafeLANG_n$? $SafeTREE_n = UnsafeTREE_n$?

Problem Status

	safe	unsafe
strictness of language hierarchy	yes [Engelfriet91]	open
strictness of tree hierarchy	yes [Damm82?]	yes [Kartzow&Parys12]
context sensitiveness	yes [Inaba&Maneth08]	open (for order ≥ 3)
safe trees = unsafe trees?	no [Parys 12]	
safe languages = unsafe languages?	open	

Outline

- Background
- Solved/open problems
- Our approach based on typed λ-calculus
  - motivation
  - pumping lemma
  - towards context-sensitiveness

Motivation

- Many results have been obtained through higher-order pushdown systems/transducers
  -> non-intuitive, complex proofs
- Simpler, more direct reasoning about grammars seems possible through λ-calculus and types

	safe	unsafe
strictness of language hierarchy	yes pushdown [Engelfriet91]	open
strictness of tree hierarchy	yes [Damm82?]	yes pushdown [Kartzow&Parys12]
context sensitiveness	yes pushdown [Inaba&Maneth08]	open (for order $\geq$ 3)

Motivation

- Many results have been obtained through higher-order pushdown systems/trandsducers
  -> non-intuitive, complex proofs
- Simpler, more direct reasoning about grammars seems possible through λ-calculus and types
  - Demonstration through:
    - pumping lemma (for deterministic case) and application to strictness of tree hierarchy
    - context-sensitiveness (ongoing work, with only preliminary result)

Outline

- Background
- Solved/open problems
- Applications of typed λ-calculus
  - motivation
  - pumping lemma for deterministic HO tree grammar (HORS)
    - background
    - statement of the lemma
    - proof sketch
  - towards context-sensitiveness

Pumping Lemmas

- State properties about "repeated structures" generated by grammars/automata

  e.g. Pumping lemma for CFL:

  "Any sufficiently long word $s \in L$ can be decomposed to

  $s = uvwxy$ (with $vx \neq \varepsilon$)

  and $uv^iwx^iy \in L$ for every $i \geq 0$"

- Used for separation of language classes

  e.g. $L = \{a^nb^nc^n \mid n \geq 0\}$ is not a CFL.

  If L were CFL, then for sufficiently large n,

  $a^nb^nc^n = uvwxy$ and $uv^iwx^iy \in L$ for every $i \geq 0$,

  but this is impossible.

Pumping lemmas for "higher-order" grammars/PDA

- Pumping lemma for indexed languages [Hayashi 73]
- Pumping lemmas for HPDS/CPDS [Parys 12; Kartzow&Parys 12]
  - strictness of hierarchy of trees/graphs generated by CPDS/HORS [Kartzow&Parys 12]
  - separation between HPDS and CPDS (or "safe" vs "unsafe" trees) [Parys 12]

Proofs are:

- complex (at least for non-experts on CPDS)
- indirect (for reasoning about HORS)

(cf. proof of pumping lemma for CFL)

Outline

- Background
- Solved/open problems
- Applications of typed λ-calculus
  - motivation
  - pumping lemma for deterministic HO tree grammar (HORS)
    - background
    - statement of the lemma and application
    - proof sketch
  - towards context-sensitiveness

Higher-Order Recursion Scheme (HORS)

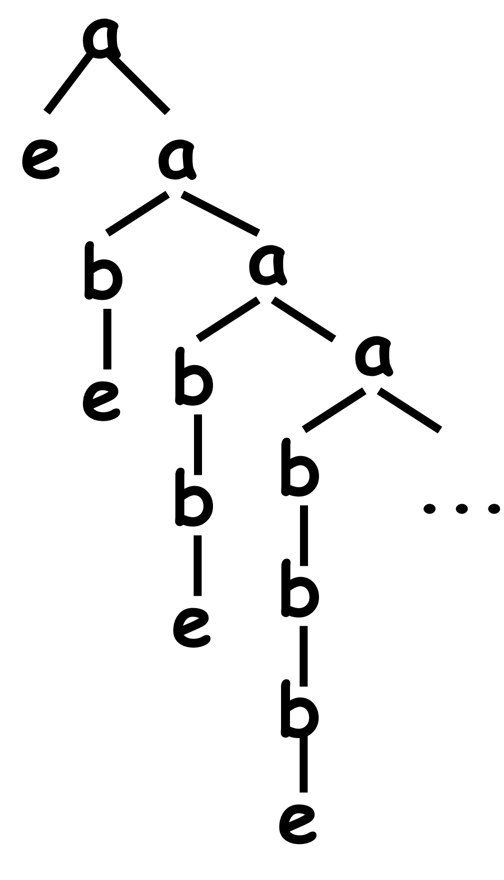
- Simply-typed, deterministic, higher-order grammar for an infinite tree

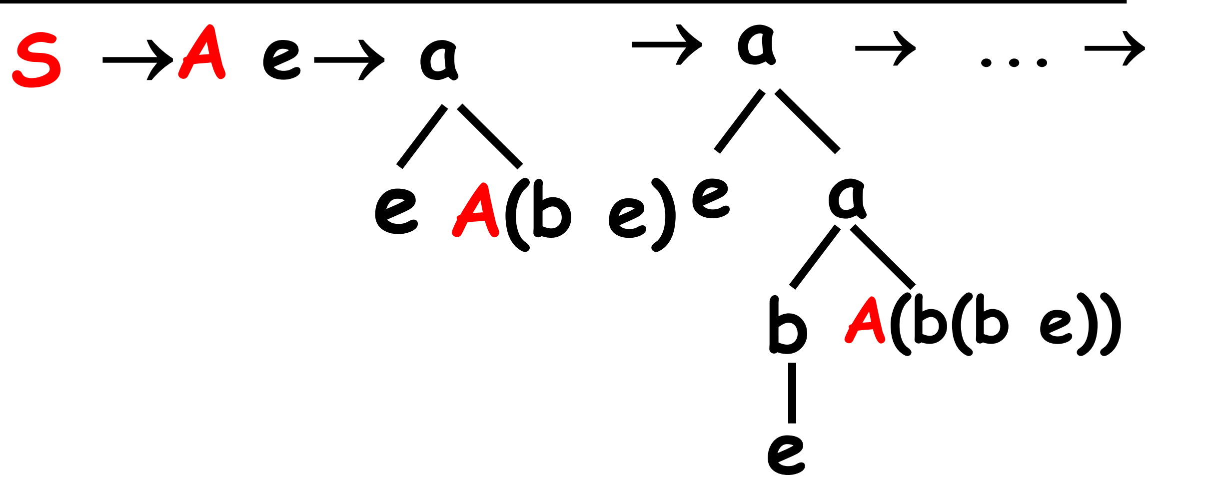
Order-1 HORS

S → A e

A x → a x (A (b x))

S: o, A: o→ o





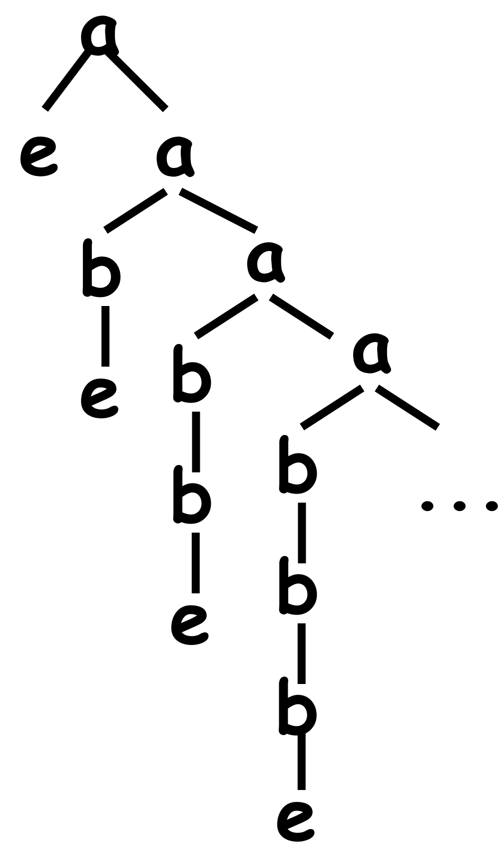
HORS as Labeled Transition System
[Carayol&Serre, LICS12]

$S \rightarrow A\ e$

$A\ x \rightarrow a\ x\ (A(b\ x))$

$$S \xrightarrow{\varepsilon} A\ e \xrightarrow{\varepsilon} a\ e\ (A\ (b\ e)) \xrightarrow{(a,2)} A\ (b\ e)$$

The second branch of node a has been chosen.



HORS as Labeled Transition System
[Carayol&Serre, LICS12]

$S \rightarrow A\ e$

$A\ x \rightarrow a\ x\ (A(b\ x))$

$S \xrightarrow{\varepsilon} A\ e$

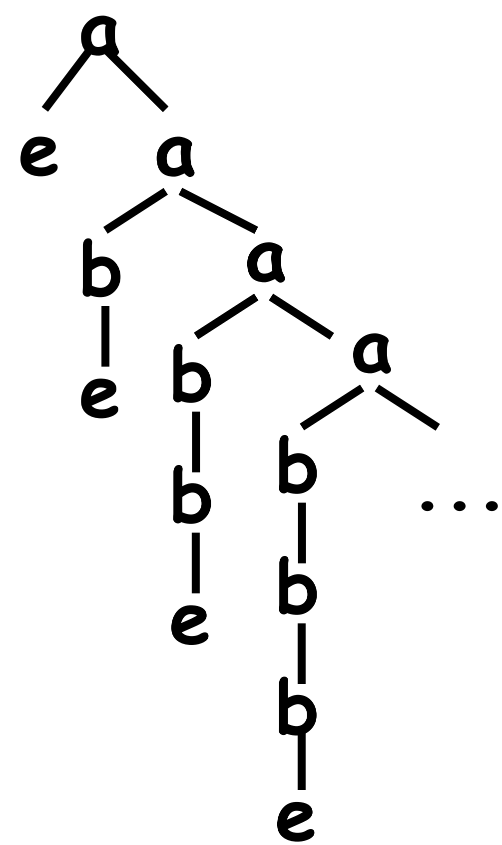
$\xrightarrow{\varepsilon} a\ e\ (A\ (b\ e))$

$\xrightarrow{(a,2)} A\ (b\ e)$

$\xrightarrow{\varepsilon} a\ (b\ e)\ (A\ (b\ (b\ e)))$

$\xrightarrow{(a,1)} b\ e$

$\xrightarrow{(b,1)} e$

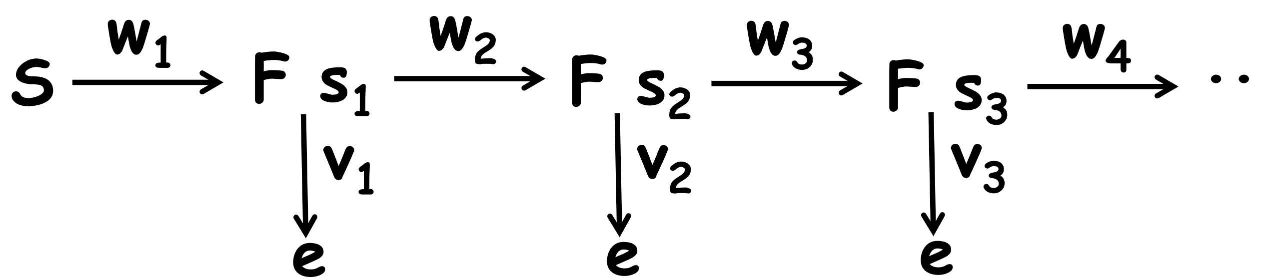


Pumping Lemma

$\forall G$: order-n HORS. $\exists c,d$.

$S \xrightarrow{w} e$ and $|w| > \exp_{n-1}(c)$ imply:

(i)



(ii) $|u_m| \leq \exp_{n-1}((m+1)c^2)$ for $u_m = w_1 \ldots w_m v_m$

(iii) $u_m \neq u_{m'}$ if $|m-m'| \geq d$

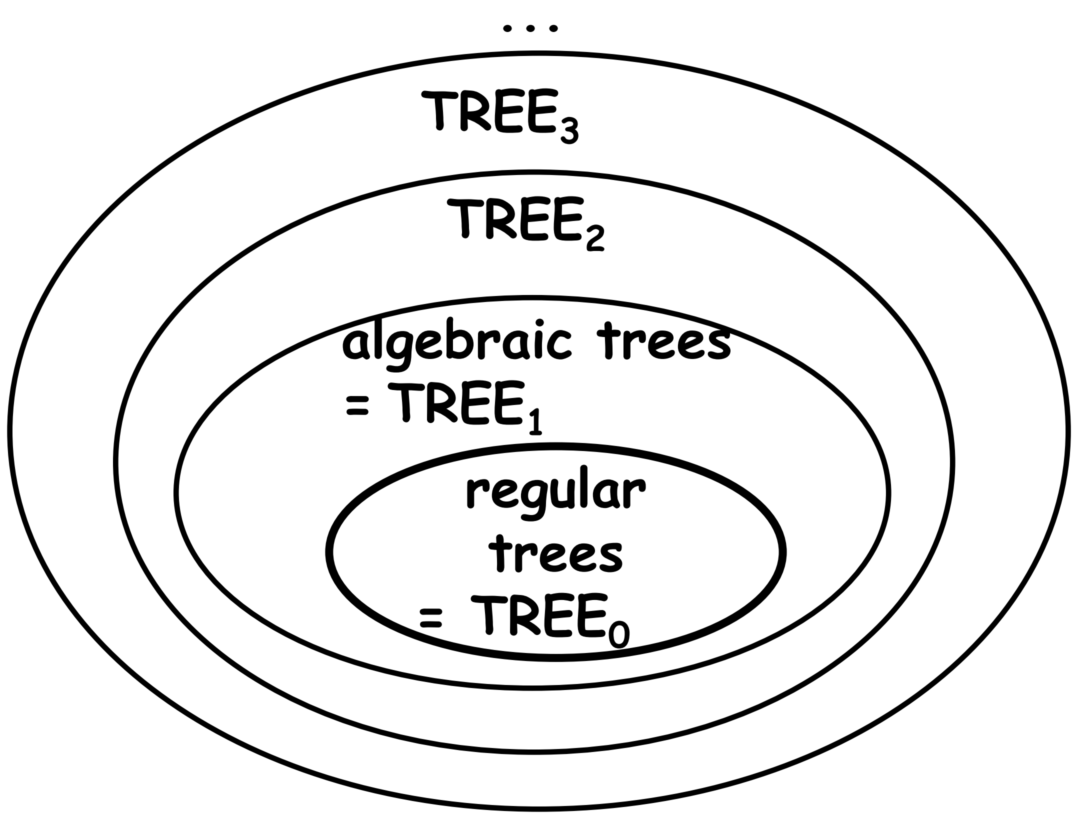
$\exp_n(x) = \underbrace{2^{2^{\cdot^{\cdot^{2^x}}}}}_{n}$

Strictness of HORS Tree Hierarchy

TREE_n = { Tree(G) | G: an order-n HORS}

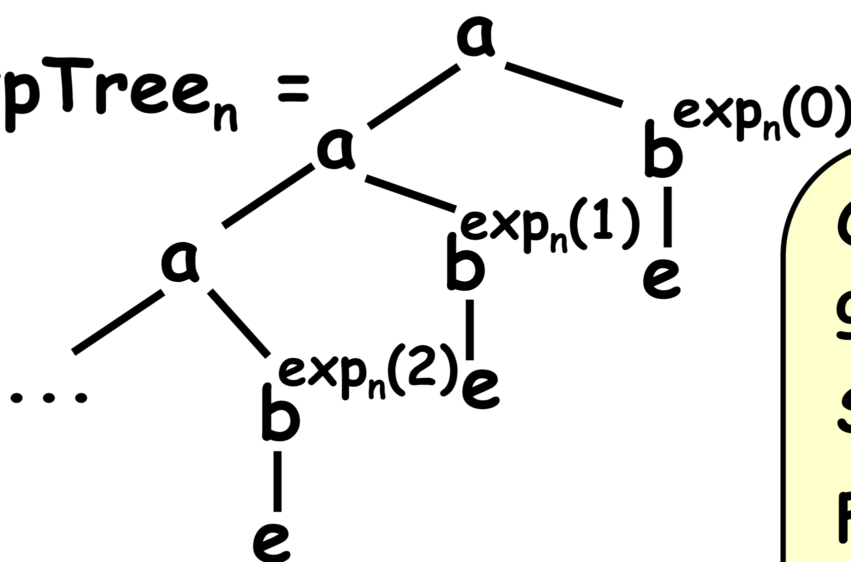
Theorem [Kartzow&Parys 12] :

$\mathrm{TREE}_0 \subsetneq \mathrm{TREE}_1 \subsetneq \ldots \subsetneq \mathrm{TREE}_n \subsetneq \mathrm{TREE}_{n+1} \subsetneq \ldots$



Witness of $\mathrm{TREE}_n \subsetneq \mathrm{TREE}_{n+1}$

$\mathrm{ExpTree}_n$ =



Order-(n+1) HORS that generates $\mathrm{ExpTree}_n$:

$S \rightarrow F\ T_{n-1}$

$F\ f \rightarrow a\ (F\ (T_n\ f))$
$\qquad (f\ T_{n-2}\ \ldots\ T_1\ b\ e)$

$T_k\ f\ x \rightarrow f(f\ x)$
$\qquad$ (for $k=1,\ldots,n$)

$$S \rightarrow F\ T_{n-1} \xrightarrow{(a,1)^k} F\ (T^k{}_n\ T_{n-1}) \xrightarrow{(a,2)} T^k{}_n\ T_{n-1} \ldots T_1\ b\ e \xrightarrow{(b,1)^{exp_n(k)}} e$$

$\text{ExpTree}_n \notin \text{TREE}_n$

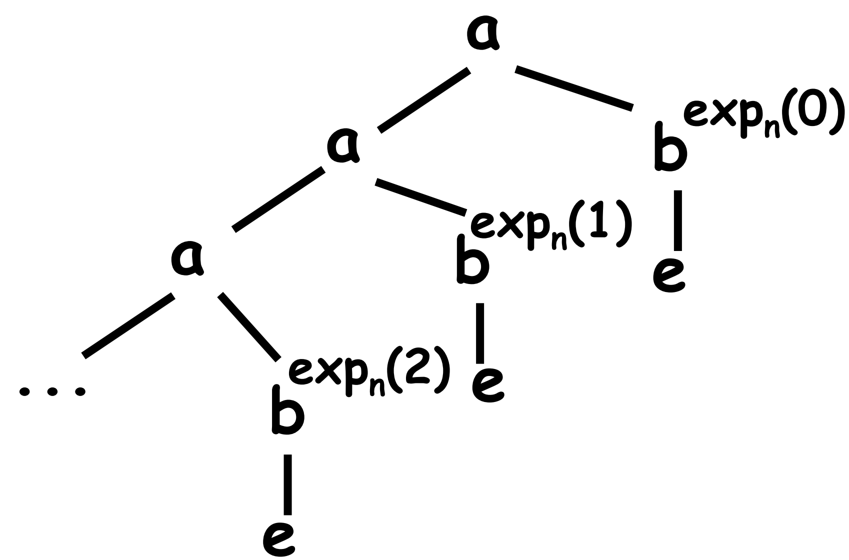
Suppose $\text{ExpTree}_n \in \text{TREE}_n$.

$\exists\ u_1, u_{1+d}, u_{1+2d}, \ldots$

(i) $S \xrightarrow{u_{1+kd}} e$

(ii) $|u_{1+kd}| \leq \exp_{n-1}((kd+2)c^2)$

(iii) $u_{1+kd} \neq u_{1+jd}$ if $k \neq j$.



$S \xrightarrow{w} e$ and $|w| > \exp_{n-1}(c)$ imply:

(i) $S \xrightarrow{w_1} F\ s_1 \xrightarrow{w_2} F\ s_2 \xrightarrow{w_3} F\ s_3 \xrightarrow{w_4} \cdots$, with $F\ s_1 \xrightarrow{v_1} e$, $F\ s_2 \xrightarrow{v_2} e$, $F\ s_3 \xrightarrow{v_3} e$

(ii) $|u_m| \leq \exp_{n-1}((m+1)c^2)$ for $u_m = w_1 \ldots w_m v_m$

(iii) $u_m \neq u_{m'}$ if $|m-m'| \geq d$

$ExpTree_n \notin TREE_n$

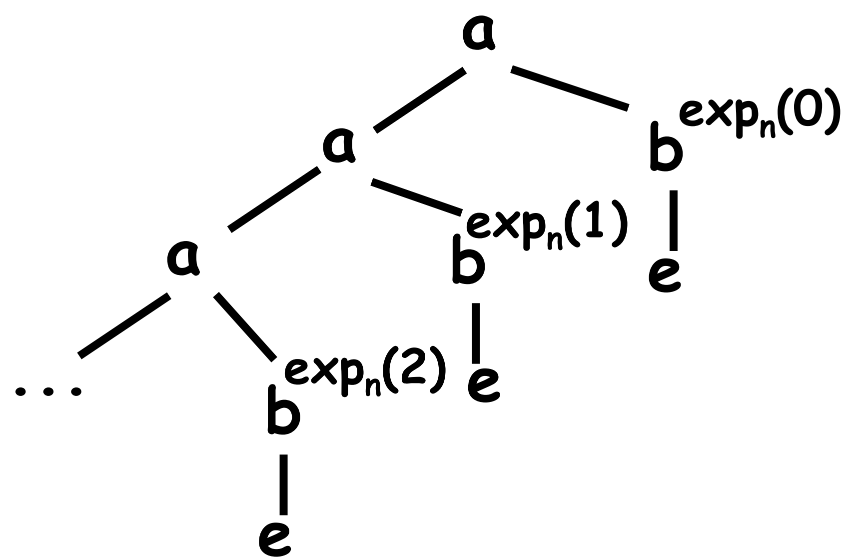
Suppose $ExpTree_n \in TREE_n$.

$\exists\ u_1,\ u_{1+d},\ u_{1+2d}, \ldots$

(i) $S \xrightarrow{u_{1+kd}} e$

(ii) $|u_{1+kd}| \leq exp_{n-1}((kd+2)c^2)$

(iii) $u_{1+kd} \neq u_{1+jd}$ if $k \neq j$.



$exp_{n-1}((kd+2)c^2)$

$\geq \max(|u_1|,\ |u_{1+d}|, \ldots,\ |u_{1+kd}|)$

$\geq$ the length of $(k+1)$-th shortest path

$= k+2+exp_n(k)$

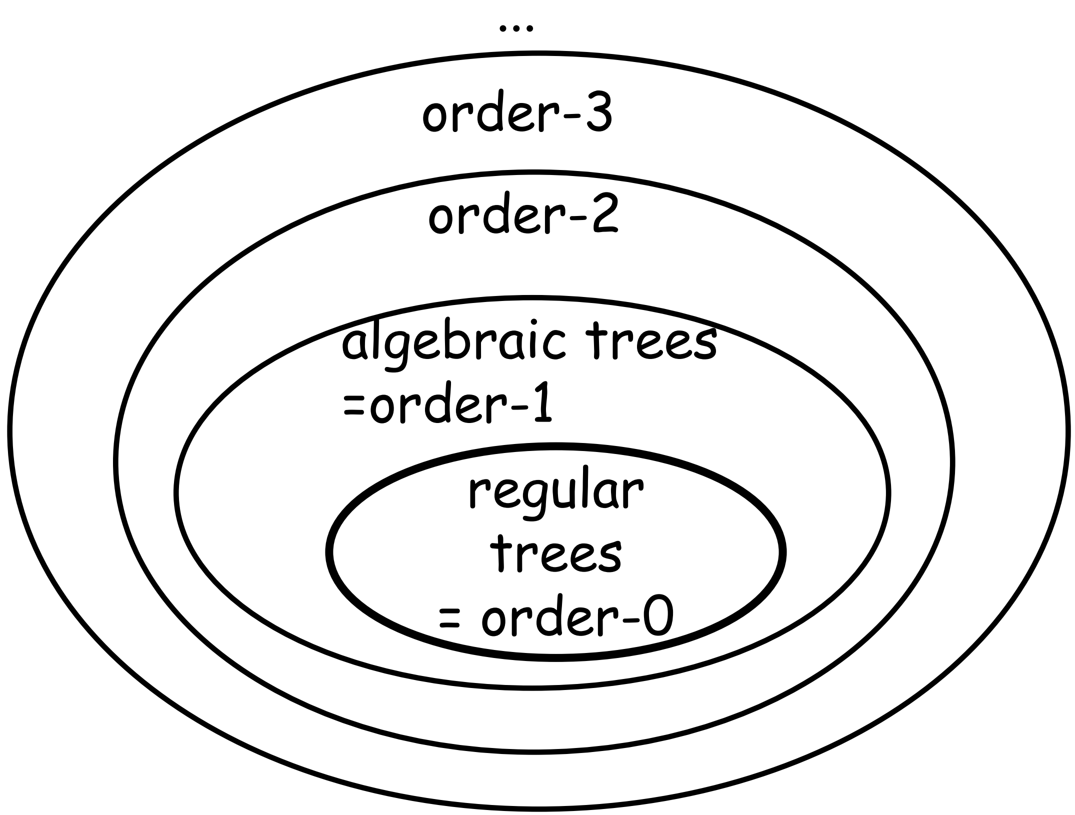
Contradiction for large k

Strictness of HORS Tree Hierarchy

TREE_n = { Tree(G) | G: an order-n HORS}

Theorem [Kartzow&Parys 12] :

$\mathrm{TREE}_0 \subsetneq \mathrm{TREE}_1 \subsetneq \ldots \subsetneq \mathrm{TREE}_n \subsetneq \mathrm{TREE}_{n+1} \subsetneq \ldots$



Outline

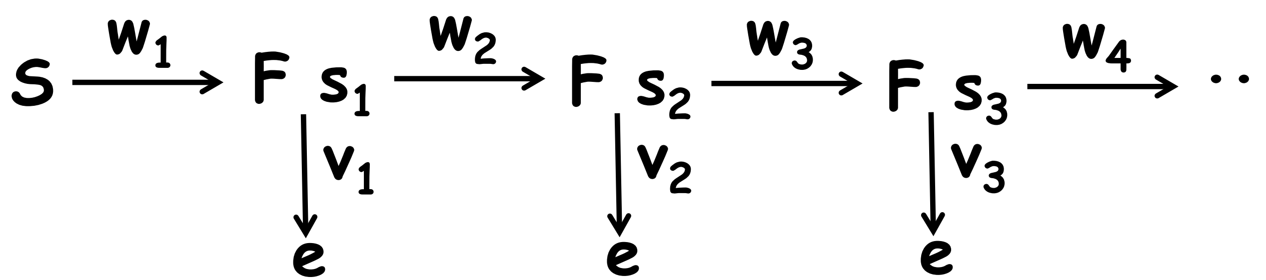
- Background
- Solved/open problems
- Applications of typed λ-calculus
  - motivation
  - pumping lemma for deterministic HO tree grammar (HORS)
    - background
    - statement of the lemma and application
    - proof sketch
  - towards context-sensitiveness

How to prove pumping lemma?

$\forall G$: order-n HORS. $\exists c,d$.

$S \xrightarrow{w} e$ and $|w| > \exp_{n-1}(c)$ imply:

(i)



(ii) $|u_m| \leq \exp_{n-1}((m+1)c^2)$ for $u_m = w_1 \ldots w_m v_m$

(iii) $u_m \neq u_{m'}$ if $|m-m'| \geq d$

$\exp_n(x) = 2^{2^{\cdot^{\cdot^{2^x}}}}$ (n 2's)

Pumping Lemma for CFL

"Any sufficiently long word $s \in L$ can be decomposed to
$s = uvwxy$ (with $vx \neq \varepsilon$)
and $uv^iwx^iy \in L$ for every $i \geq 0$"

Proof.

The derivation of s must contain repeated occurrences of a non-terminal F:

$S \to^* uFy \to^* uvFxy \to^* uvwxy$ (=s).

By repeating the part $F \to^* vFx$,

$S \to^* uFy \to^* uvFxy \to^* uvvFxxy \to^* uv^iFx^iy \to^* uv^iwx^iy$

Pumping for HORS?

- Sufficiently long transition sequence must contain repeated occurrences of non-terminal in the head position?

$$S \xrightarrow{u} F\ s_1 \xrightarrow{w} F\ s_2 \xrightarrow{v} e$$

Yes!

- Can the part "$F\ s_1 \rightarrow F\ s_2$" be pumped??

Not necessarily:

For $G = \{S \rightarrow F(F\ e),\ F\ x \rightarrow x\}$,

$S \rightarrow F(F\ e) \rightarrow F\ e \rightarrow e$

but the part "$F(F\ e) \rightarrow F\ e$" cannot be repeated!

Conditions for Pumping F $s_1 \to$ F s_2?

- F should be obtained by unfolding F
  - × F(F e) → F e where F x → x
  - ✓ F e $\to^{(a,1)}$ F (c e) where F x → a (F (c x))

  Sufficient?

  No.

  For G={S→F I, F f→f(F K), I x→x, K x→e},

  S → F I → I(F K) → F K,

  but F K → K(F K) → e $\not\to^*$ F ...

  because I and K have different behaviors!

  (I uses the argument but K ignores it)

Conditions for Pumping F $s_1 \to$ F s_2?

- F should be obtained by unfolding F
  - × F(F e) → F e where F x → x
  - ✓ F e $\to^{(a,1)}$ F (c e) where F x → a (F (c x))
- s_2 should have the same type as s_1

  For G={S→F I, F f→f(F K), I x→x, K x→e},

  S → F I → I(F K) → F K,

  but I: r→r and K: T→r

Use an argument

Ignore an argument

Intersection Types for Expressing Reduction Behavior

τ (types) ::=

r (terms that reduce to e)

$(\tau_1 \wedge \ldots \wedge \tau_k) \to \tau$ (functions that use an argument as a value of types $\tau_1, \ldots, \tau_k$ and return a value of type τ)

Types	Examples	Non-examples
$r \to r$	$\lambda x.x$, $\lambda x.a\ x\ e$	$\lambda x.e$
$T \to r$	$\lambda x.e$, $\lambda x.a\ x\ e$	$\lambda x.x$
$(r \to r) \to r$	$\lambda f.f\ e$, $\lambda f.f(f\ e)$	$\lambda f.e$

(inaccurate) Key Lemma

If

(i) F $s_1 \rightarrow^* $ F s_2

(ii) F comes from F

(iii) F and F have the same type τ,

then

(1) F $s_2 \rightarrow^* $ F s_3 for some s_3

(2) F comes from F

(3) F has type τ

Proof Sketch of Pumping Lemma

1. A sufficiently long transition sequence is of the form (for a sufficiently large k > #types):

$$S \xrightarrow{u_1} F^{\tau_1} t_1 \xrightarrow{u_2} F^{\tau_2} t_2 \xrightarrow{u_3} \ldots \xrightarrow{u_k} F^{\tau_k} t_k \xrightarrow{u_{k+1}} e$$

2. Assign an intersection type to each F.

3. Pick i and j such that $\tau_i = \tau_j$ and "pump" the part $F\ t_i \xrightarrow{u_{i+1} \ldots u_j} F\ t_j$

$S \xrightarrow{w} e$ and $|w| > \exp_{n-1}(c)$ imply:

(i) $S \xrightarrow{w_1} F\ s_1 \xrightarrow{w_2} F\ s_2 \xrightarrow{w_3} F\ s_3 \xrightarrow{w_4} \cdots$, with $F\ s_1 \xrightarrow{v_1} e$, $F\ s_2 \xrightarrow{v_2} e$, $F\ s_3 \xrightarrow{v_3} e$

(ii) $|u_m| \leq \exp_{n-1}((m+1)c^2)$ for $u_m = w_1 \ldots w_m v_m$

(iii) $u_m \neq u_{m'}$ if $|m-m'| \geq d$

Proof Sketch

1. A sufficiently long transition sequence is of the form (for a sufficiently large k > #types):

$$S \xrightarrow{u_1} F^{\tau_1}\, t_1 \xrightarrow{u_2} F^{\tau_2}\, t_2 \xrightarrow{u_3} \cdots \xrightarrow{u_k} F^{\tau_k}\, t_k \xrightarrow{u_{k+1}} e$$

2. Assign an intersection type to each F.

3. Pick i and j such that $\tau_i = \tau_j$ and "pump" the part $F\ t_i \xrightarrow{u_{i+1} \ldots u_j} F\ t_j$ and obtain:

$$S \xrightarrow{w_1} F\ s_1 \xrightarrow{w_2} F\ s_2 \xrightarrow{w_3} F\ s_3 \xrightarrow{w_4} \cdots$$

$$F\ s_1 \xrightarrow{v_1} e \qquad F\ s_2 \xrightarrow{v_2} e \qquad F\ s_3 \xrightarrow{v_3} e$$

Proof Sketch

...

3. Pick i and j such that $\tau_i = \tau_j$ and "pump" the part $F\ t_i \xrightarrow{u_{i+1}\ldots u_j} F\ t_j$ and obtain:

$$\begin{array}{ccccccccc} S & \xrightarrow{w_1} & F\ s_1 & \xrightarrow{w_2} & F\ s_2 & \xrightarrow{w_3} & F\ s_3 & \xrightarrow{w_4} & \cdots \\ & & \downarrow v_1 & & \downarrow v_2 & & \downarrow v_3 & & \\ & & e & & e & & e & & \end{array}$$

4. To obtain the bound $|w_1\ldots w_m v_m| \leq \exp_{n-1}((m+1)c^2)$, simulate $S \xrightarrow{w_1\ldots w_m v_m} e$

   by: $t \xrightarrow{w_1\ldots w_m v_m} e$

   for a λ-term t (obtained by unfolding S). $|w_1\ldots w_m v_m|$ is bounded by the size of β-normal form of t [Beckmann 01]

Pumping Lemma for HORS: summary

- The same reasoning as for context-free languages is possible, with a help of types
  - A sufficiently long transition sequence must contain repeated occurrences of the same non-terminal
  - The part where the same non-terminal is used as the same type can be pumped
  - The length of pumped words can be bounded by using the standard result on the size of β-normal form of simply typed λ-terms

Pumping Lemma for Word Language Grammar?

$S \xrightarrow{w} e$ and $|w| > \exp_{n-1}(c)$ imply:

(i) $$\begin{array}{ccccccccc} S & \xrightarrow{w_1} & F\ s_1 & \xrightarrow{w_2} & F\ s_2 & \xrightarrow{w_3} & F\ s_3 & \xrightarrow{w_4} & \cdots \\ & & \downarrow v_1 & & \downarrow v_2 & & \downarrow v_3 & & \\ & & e & & e & & e & & \end{array}$$

(ii) $|u_m| \leq \exp_{n-1}((m+1)c^2)$ for $u_m = w_1 \ldots w_m v_m$

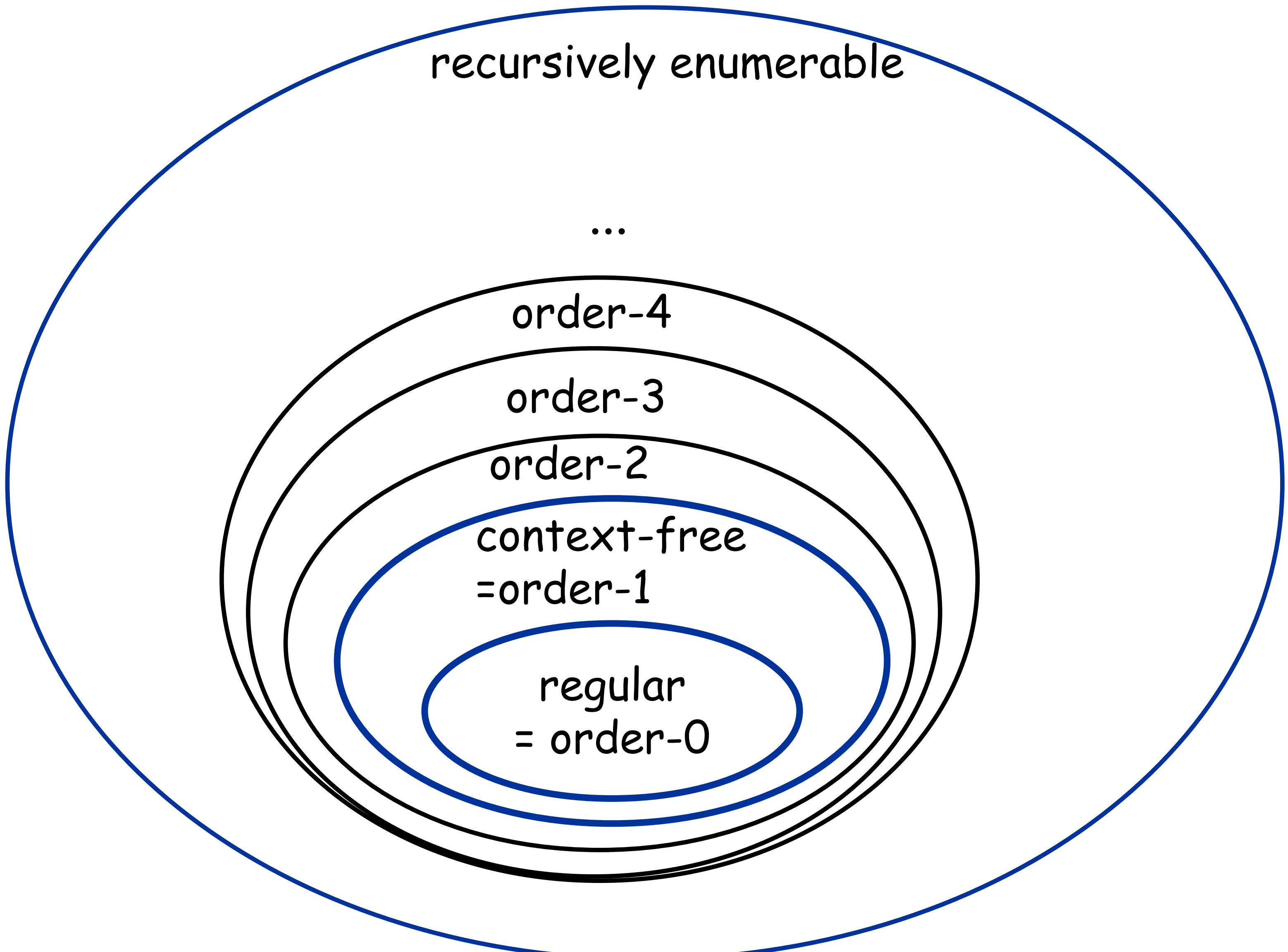
(iii) $u_m \neq u_{m'}$ if $|m-m'| \geq d$

→ A further twist is required

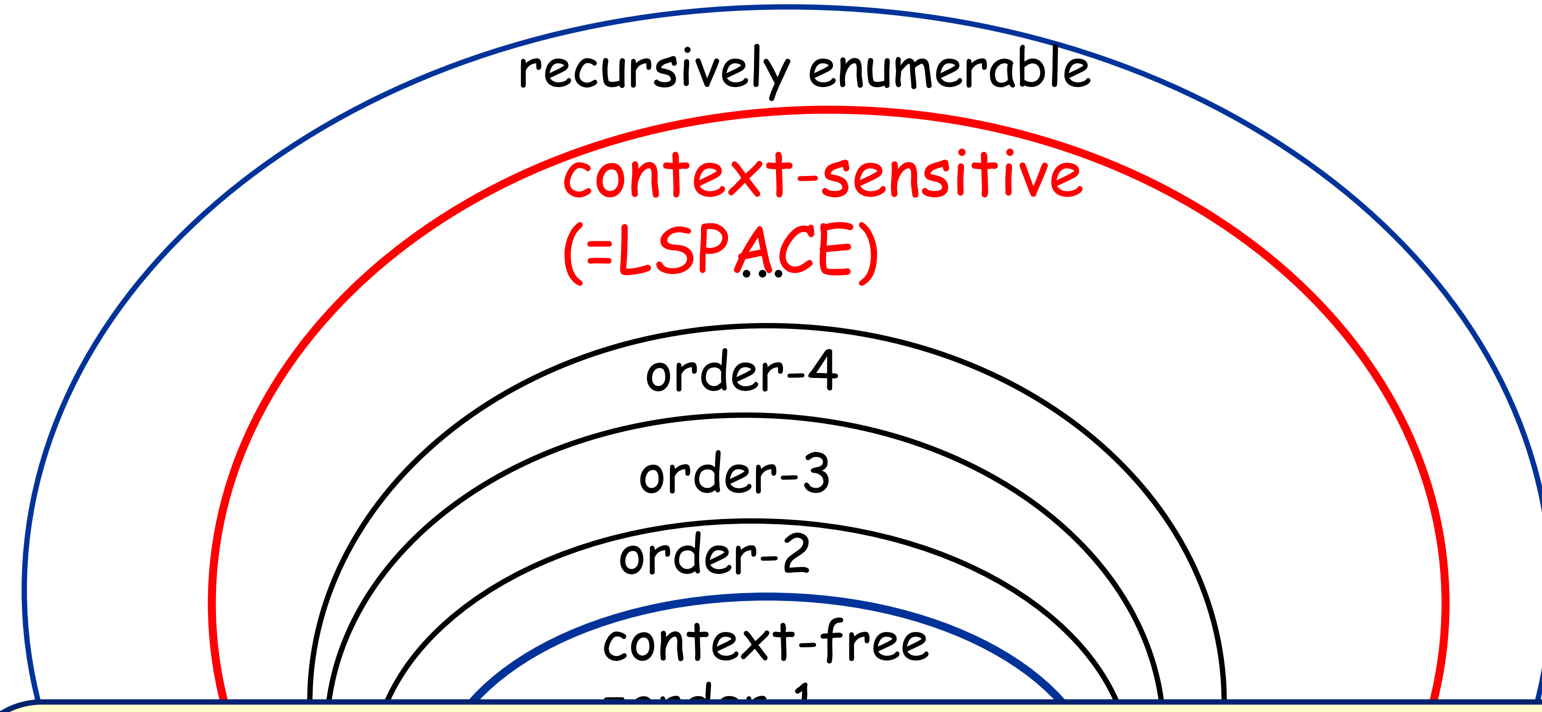
Outline

- Background
- Solved/open problems
- Applications of typed λ-calculus
  - motivation
  - pumping lemma
  - towards context-sensitiveness of unsafe languages (ongoing work with Kazuhiro Inaba and Takeshi Tsukada)

Chomsky hierarchy and HO languages



Chomsky hierarchy and HO languages



Theorem [Inaba&Maneth 08]
Safe order-n word languages are context-sensitive for every n
(The proof goes through decomposition of higher-order transducers.)

Context-sensitiveness of Context-free languages

- Eliminate ε-generating rules

S -> aAb
A -> ε   A -> aAb

↓

S -> aAb   S -> ab
A -> ε   A -> aAb   A -> ab

The normalized grammar has only monotonically increasing production sequences

($S \rightarrow t_1 \rightarrow t_2 \rightarrow \ldots \rightarrow w$ implies $|S| \leq |t_1| \leq |t_2| \leq \ldots \leq |w|$)

$\Rightarrow$ membership is NLINSPACE

Context-sensitiveness of Order-1 tree grammar?

- What should be removed to ensure the monotonicity of production sequence?
  - Redundant arguments

$S \to F\ e$	$F\ x \to e$	$F\ x \to F\ (a\ x)$

$S \to F\ e \to F\ (a\ e) \to^* F\ (a^n\ e) \to e$

$S \to F\ e$	$F\ x \to e$	$F\ x \to F\ (a\ x)$

cf. type-based useless code elimination [Damiani&Prost 96, K 2000]

Context-sensitiveness of Order-1 tree grammar?

- What should be removed to ensure the monotonicity of production sequence?
  - Redundant arguments

$S \to F\ e \qquad F\ x \to e \qquad F\ x \to F\ (a\ x)$

  - Identity functions

$S \to F\ e \qquad F\ x \to x \qquad F\ x \to F\ (F\ x)$

$S \to F\ e \to F\ (F\ e) \to F^n\ e \to F\ e \to e$

$S \to e \mid F\ e \qquad F\ x \to x \qquad F\ x \to F\ (F\ x) \mid F\ x$

Context-sensitiveness of Order-2 tree grammar?

- What should be removed to ensure the monotonicity of production sequence?
  - Redundant arguments
  - Identity functions
  - Permutators (+α)

$S \to F\ a \qquad F\ x \to x\ c\ d \qquad F\ x \to F\ (G\ x)$
$G\ x\ y\ z \to x\ z\ y$

$S \to F\ a \to F\ (G\ a) \to^* F\ (G^{2n}\ a) \to (G^{2n}\ a)\ c\ d$
$\to G^{2n-1}\ a\ d\ c \to G^{2n-2}\ a\ c\ d \to^*\ a\ c\ d$

$S \to F\ a \qquad F\ x \to x\ c\ d \qquad F\ x \to F_G\ x$
$F_G\ x \to x\ d\ c \qquad F_G\ x \to F\ x \qquad G\ x\ y\ z \to x\ z\ y$

Open Problems (for proving context-sensitiveness)

- What should be removed to ensure the monotonicity of production sequence for grammars of arbitrary orders?
  - Removing hereditary permutators is necessary, but not sufficient
- Is there a systematic transformation that removes them?

A positive answer would imply context-sensitiveness of (unsafe) higher-order languages

Summary

- A survey of properties of higher-order languages
  - Many problems have been solved for safe languages, but open for unsafe ones
- λ-calculus and types seem to be a promising approach to studies of unsafe languages
  - simpler proof of strictness of tree hierarchy
  - work is under way for proving context-sensitiveness of HO languages
  - "safety" is natural for HPDS,
    but "unsafety" is more natural for λ-calculus