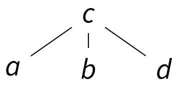
ν-Tree Languages

Leonhard Staut
Advisor: Dominik Kirst
Supervisor: Prof. Dr. Gert Smolka
Saarland University
September 20, 2017

Recap

- Tree languages over infinite alphabet $\mathbb{A}$ undecidable in general
- Class of tree languages with similar structure arising from systematic permutation of names

$\Big\{$ 

$\Big|\ a, b \in \mathbb{A} \setminus \{d, c\}; a \neq b \ \Big\}$

- Finitary representation: ν-trees

Thesis summary

- Formalization of ν-trees and their language $[\![-]\!]$
- Decidability of $[\![-]\!]$
- Equivalence laws for $[\![-]\!]$
- Decidable ν-tree automaton model

ν-Tree

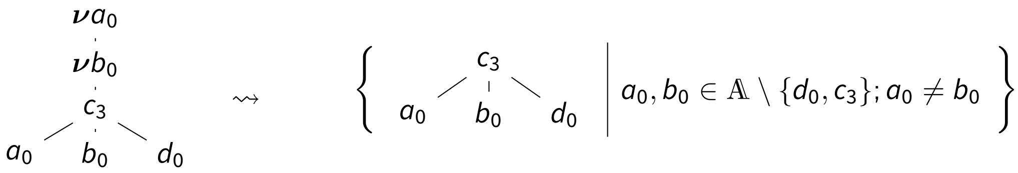
Definition (ν-Tree [Kirst, 2016])

The type ν-Tree is defined inductively by

$$n ::= a_k n_1 \ldots n_k \mid \nu a_k . n$$

where a_k ranges over the enumerable ranked alphabet $\mathbb{A}$.

- Language $[\![n]\!]$ is a class of pure trees with
  - Same structure
  - Instantiated ν-bindings with fresh names



ν-Tree Language

Definition (ν-Tree Language)

$$\frac{t_i \in [\![n_i]\!]_{a_k::A}}{a_k t_1 \ldots t_k \in [\![a_k n_1 \ldots n_k]\!]_A}$$

$$\frac{t \in [\![(a_k b_k) \cdot n]\!]_{b_k::A} \qquad b_k \notin A \quad b_k \notin \mathtt{FN}(\boldsymbol{\nu} a_k.n)}{t \in [\![\boldsymbol{\nu} a_k.n]\!]_A}$$

- $(a_k b_k)$ is the transposition of a_k and b_k
- $\mathtt{FN}(n)$ are the free names in n
- $$\pi \cdot (a_k n_1 \ldots n_k) = (\pi a_k)(\pi \cdot n_1) \ldots (\pi \cdot n_k)$$ $$\pi \cdot (\boldsymbol{\nu} a_k.n) = \boldsymbol{\nu}(\pi a_k).(\pi \cdot n)$$
- Equivariance: $\pi \cdot [\![n]\!]_A \equiv [\![\pi \cdot n]\!]_{\pi \cdot A}$

ν-Tree Language Equivalence Laws

- Laws of the form $[\![n]\!]_A \equiv [\![n']\!]_A$
  - For two ν-trees we also write $n \equiv n' := \forall A.\ [\![n]\!]_A \equiv [\![n']\!]_A$
- First step towards future work on a decision procedure for $[\![n]\!]_A \equiv [\![n']\!]_A$
- Nominal axioms for ν-words hold for ν-tree
- Nominal axioms as fragment of the nominal Kleene algebra [Gabbay and Ciancia, 2011]

$$
\begin{aligned}
b \notin \mathtt{FN}(x) \rightarrow \nu a.x &= \nu b.\ (ab) \cdot x \\
\nu a.\nu b.x &= \nu b.\nu a.x \\
a \notin \mathtt{FN}(x) \rightarrow \nu a.x &= x \\
a \notin \mathtt{FN}(x) \rightarrow x(\nu a.y) &= \nu a.xy
\end{aligned}
$$

General Renaming

Theorem

π fixes $\mathtt{FN}(n) \rightarrow [\![n]\!]_A \equiv [\![\pi \cdot n]\!]_A$

- Characteristic property for ν-tree languages
- Not a nominal axiom
- Proof by induction
  - Tree case easy
  - In the case of a binding νa_k we have an instantiation b_k, such that $t \in [\![(a_k b_k) \cdot n]\!]_{b_k::A}$
  - Show that b_k is also the right instantiation for $\nu(\pi a_k)$ by rewriting permutations

Nominal axiom: Renaming of ν-Bindings

Theorem

$b_k \notin \mathtt{FN}(\nu a_k.n) \rightarrow [\![\nu a_k.n]\!]_A \equiv [\![\nu b_k.\,(a_k b_k) \cdot n]\!]_A$

$$\begin{array}{c} c_1 \\ | \\ \nu a_0 \\ | \\ a_0 \end{array} \quad \equiv \quad \begin{array}{c} c_1 \\ | \\ \nu b_0 \\ | \\ b_0 \end{array}$$

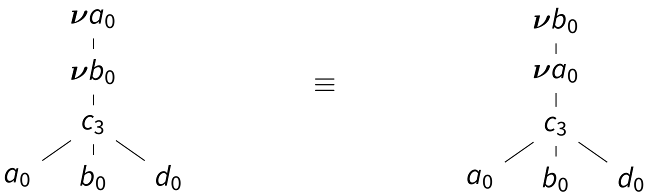
- Instance of general renaming
- $a_k \notin \mathtt{FN}(\nu a_k.n)$ and $b_k \notin \mathtt{FN}(\nu a_k.n)$
- $(a_k b_k)$ is a renaming

Nominal axiom: Swapping of ν-Bindings

Theorem

$[\![\nu a_k.\nu b_l.n]\!]_A \equiv [\![\nu b_l.\nu a_k.n]\!]_A$

- No conflicts when instantiating successive ν-bindings



- Proof idea: Show that any instantiation in the left ν-tree is a valid instantiation in the right ν-tree
- Show that the freshness conditions stay the same when swapping

Weakening and Strengthening for $[\![n]\!]_A$

- List A carries names that may not be used to instantiate bindings
- Weakening removes names from A, strengthening adds names to A

Lemma (Weakening)

$t \in [\![n]\!]_{c::A} \to t \in [\![n]\!]_A$

Lemma (Strengthening)

$t \in [\![n]\!]_A \to c \notin \mathtt{Name}(t) \to t \in [\![n]\!]_{c::A}$

- Only names not used for instantiation may be added to A
- Proof by induction on $[\![-]\!]$
- Use that instantiations have to appear in the tree t

Nominal axiom: Empty ν-Bindings

Theorem

$a_k \notin \mathtt{FN}(n) \rightarrow [\![\nu a_k.n]\!]_A \equiv [\![n]\!]_A$

- Significant equivalence for decidability of $t \in [\![n]\!]_A$

$$b_0 \quad \equiv \quad \begin{array}{c} \nu a_0 \\ | \\ b_0 \end{array}$$

- Proof by Renaming and Weakening/Strengthening

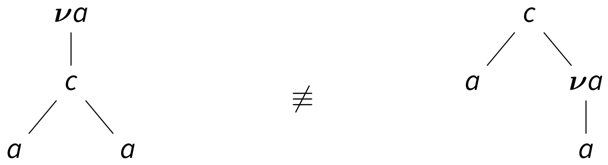
$$\begin{array}{ll}
t \in [\![\nu a_k.n]\!]_A & \\
t \in [\![(a_k b_k) \cdot n]\!]_{b_k::A} & \textit{Definition} \\
t \in [\![n]\!]_{b_k::A} & \textit{Renaming} \\
t \in [\![n]\!]_A & \textit{Weakening}
\end{array}$$

Nominal axiom: Pushing down ν-Bindings

- Change position of ν-binding
- Push ν-binding along a path
  - Identify unique subtree n_j to push the binding to

Binding positioning

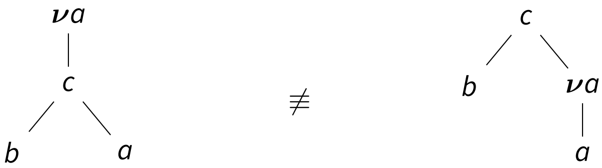
- Names in scope depend on position



- Cannot re-position binding if scope is changed

Binding positioning

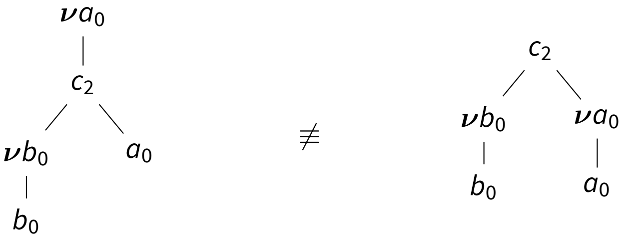
- Freshness conditions imposed by free names depend on position



- Cannot re-position binding if visibility of free names is changed

Binding positioning

- Freshness conditions imposed by other ν-bindings depend on position



- Cannot re-position binding if visibility of other bindings is changed

Nominal Axiom: Pushing $\boldsymbol{\nu}$-Bindings (ctd.)

- Let n_j be the subtree where the ν-binding is placed

$$(\forall\, l \neq j.\, a_k \notin \mathtt{FN}(n_l)) \quad \text{“Scope invariance”}$$

$$\rightarrow \mathtt{FN}(\boldsymbol{\nu} a_k.c_k(n_1 \ldots n_j \ldots n_k)) \setminus \{A\} \subseteq \mathtt{FN}(\boldsymbol{\nu} a_k.n_j) \quad \text{“}\mathtt{FN}\text{ invariance”}$$

$$\rightarrow (\forall\, l \neq j.\, \nexists\, (\boldsymbol{\nu} d_k.n') \in n_l) \quad \text{“}\boldsymbol{\nu}\text{ invariance”}$$

$$\rightarrow [\![\boldsymbol{\nu} a_k.c_k(n_1 \ldots n_j \ldots n_k)]\!]_A \equiv [\![c_k(n_1 \ldots (\boldsymbol{\nu} a_k.n_j) \ldots n_k)]\!]_A$$

- First assumption necessary because of scoping
- Second and third because of freshness conditions

Future work

- Formalization of decision procedure for $[\![n]\!] \equiv [\![n']\!]$ using the equivalence laws
  - Remove empty ν-bindings
  - Push remaining ν-bindings down
  - If equivalent, normalized ν-trees are equal up to names in bindings
  - Equality up to bound names decidable
- Decidability of emptiness for NTA languages
- Complement of NTA

References

- Gabbay, M. K. and Ciancia, V. (2011). Freshness and name-restriction in sets of traces with names. FOSSACS'11/ETAPS'11, Berlin, Heidelberg. Springer-Verlag.
- Kirst, D. (2016). Intersection type systems corresponding to nominal automata. Master's thesis, Oxford University.
- Pitts, A. M. (2013). *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press.
- Stirling, C. (2009). *Dependency Tree Automata*. Springer Berlin Heidelberg, Berlin, Heidelberg.

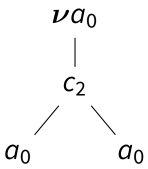
- `https://www.ps.uni-saarland.de/~staut/bachelor.php`

Appendix: Lines of code

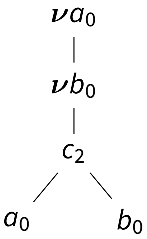
- Linear development structure

spec	proof	
228	248	Base
103	101	Name permutations
190	298	Lists
38	39	Pure trees
212	274	ν-trees
130	291	Equivalence laws
89	221	Decidability of $t \in [\![n]\!]$
173	174	NTA
1163	1646	total

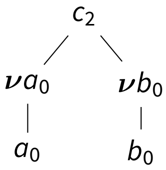
Appendix: ν-Tree Expressiveness



νa_0 instantiated with one name



νa_0 and νb_0 instantiated with two different names



νa_0 and νb_0 instantiated with two arbitrary names

Appendix: α-Equivalence for ν-trees

- Equivalent language
- Not α-equivalent, since bound names in one tree cannot be obtained from the other by permutation
- No other equivalence law is applicable