Level Two Recursion Schemes and Finite Automata

Klaus Aehlig

(joint work with Jolie de Miranda and Luke Ong)

Supported by the German Academic Exchange Service (DAAD).

The Problem

- infinite tree t, given by a level 2 recursion scheme

  (idealised functional program)

- Monadic Second Order formula φ

Question:

$$t \models \varphi \quad ?$$

. . . and make this decidable.

Propaganda

- Automatic verification is important. . .
- Functional programming is natural, expressive, . . .

(Level 2) Recursion Schemes

Short answer: simply typed lambda terms with fixed points

Long answer: A level 2 recursion scheme is given by

- non-terminals $N : (\iota \to \ldots \to \iota) \to \ldots \to \iota \to \ldots \to \iota$
  one of them: the start symbol S
- terminals $\mathtt{f} : \iota \to \ldots \to \iota$
- Equations for all non-terminals N

$$N\overrightarrow{\varphi}\overrightarrow{x} = e$$

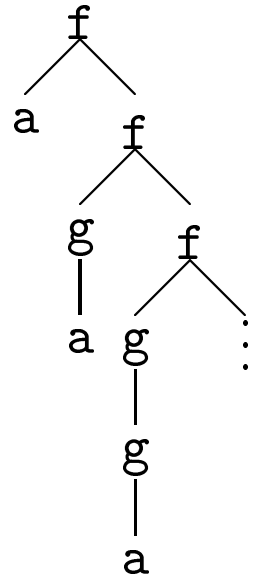
$\uparrow$

built of

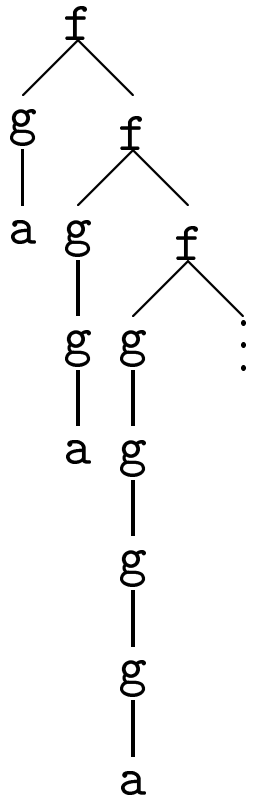
- terminals $\mathtt{f}, \mathtt{g}, \mathtt{a}, \ldots$
- variables $\overrightarrow{\varphi} : \iota \to \ldots \to \iota \to \iota$, $\overrightarrow{x} : \iota$
- arbitrary non-terminals $N, M, \ldots$

Examples

$$
\begin{aligned}
S &= F\mathtt{a} \\
Fx &= \mathtt{f}x(F(\mathtt{g}x))
\end{aligned}
$$



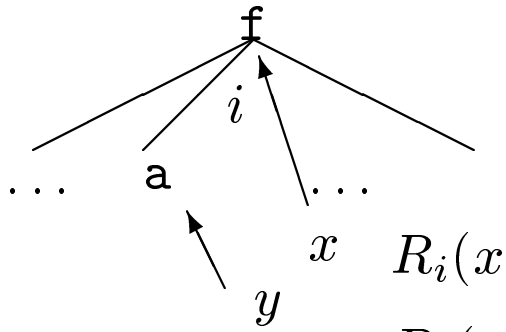
$$
\begin{aligned}
S &= F\mathtt{g} \\
B\varphi\psi x &= \varphi(\psi x) \\
F\varphi &= \mathtt{f}(\varphi\mathtt{a})(F(B\varphi\varphi))
\end{aligned}
$$



MSO Formulae...

- Child relation between nodes
- Labelling of a node
- propositional logic: $\wedge$, $\vee$, $\neg$, ...
- Quantification: $\forall x$, $\exists x$
- Sets of nodes X with new atomic formula $x \in X$ and quantification $\forall X$, $\exists X$

  like: "there is a path..."



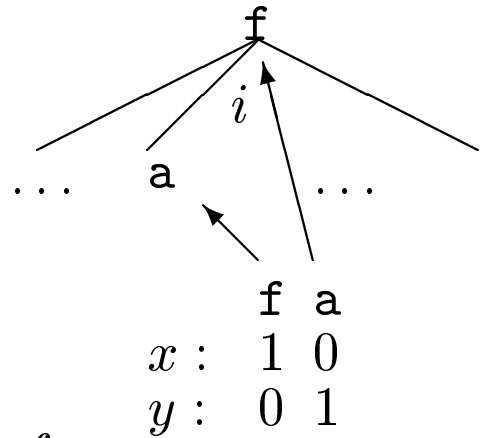
$R_i(x, y)$

$P_f(x)$ $P_a(y)$

. . . can be understood as tree automata

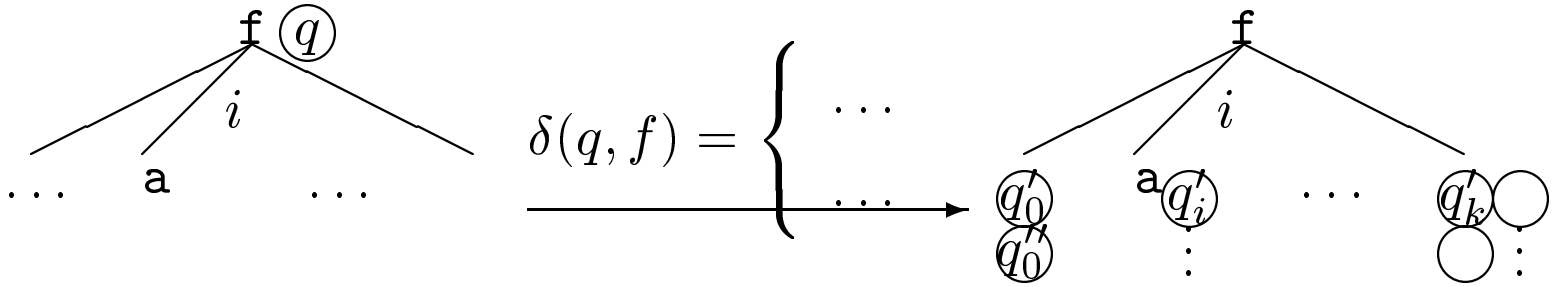
- first order variables $\longrightarrow$ singleton sets
  (and $x \in X$ *iff* $\{x\} \subset X$*)*

$\rightsquigarrow$ can be treated as additional labels

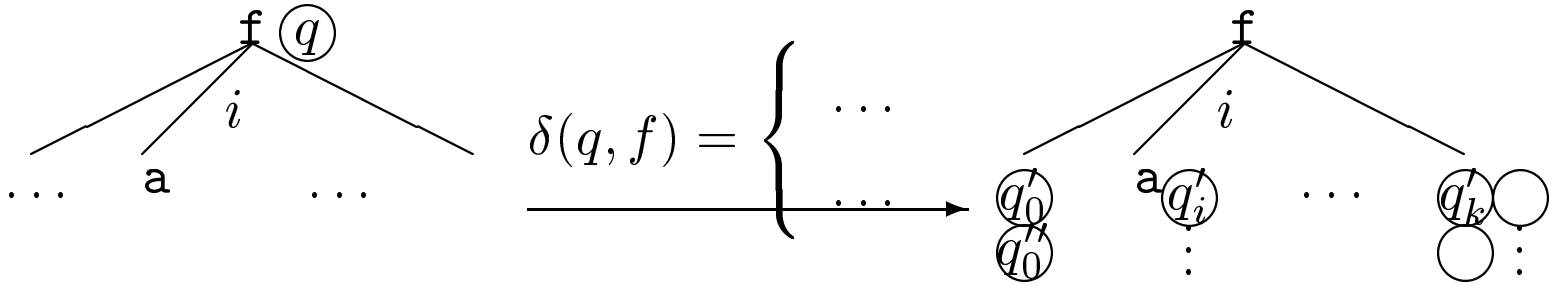


- $\exists X$ becomes projection of the alphabet

$\therefore$ validity can be understood as the successful run of an appropriate tree automaton



Tree Automata (continued)



- choices at every step ($\therefore$ projection, i.e., $\exists X$)
- but each of the new automata has to succeed

  i.e., "alternation". Choose a model of a positive boolean combination

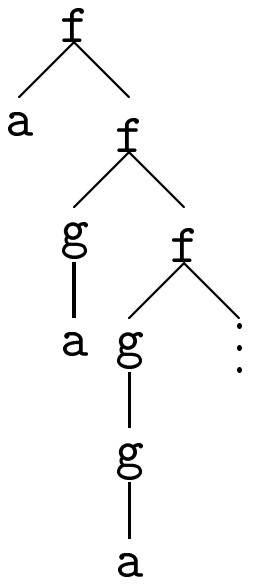
- acceptance condition by the set of states visited infinitely often on each path (requiring $\mathrm{Inf}(\pi) \in \Omega \subset \mathfrak{P}(Q)$)

$\rightsquigarrow$ closure under complement (negation!)

Essence: **MSO properties $\equiv$ languages of appropriate tree automata**

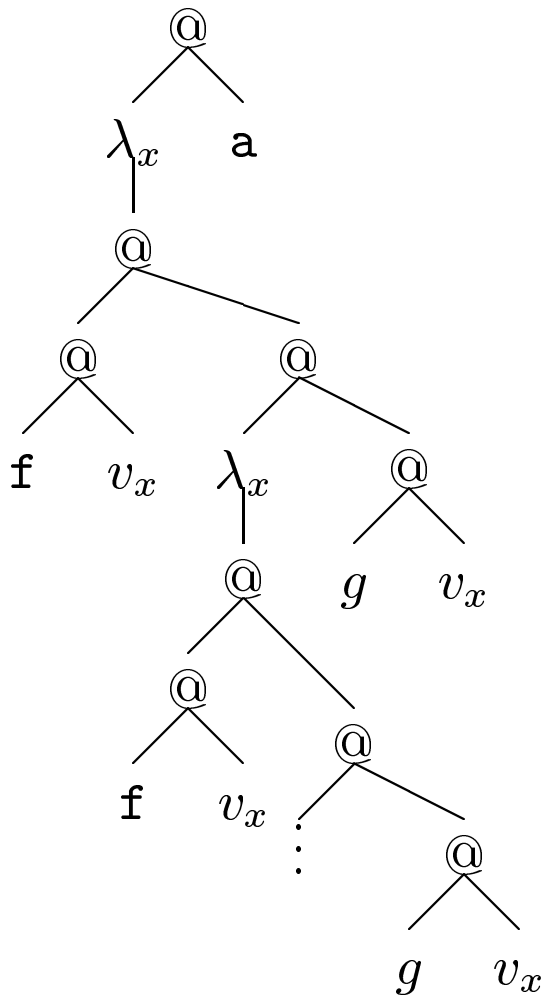
Making the trees regular

$$\begin{aligned} S &= F\mathtt{a} \\ Fx &= \mathtt{f}x(F(\mathtt{g}x)) \end{aligned}$$



$\rightsquigarrow$

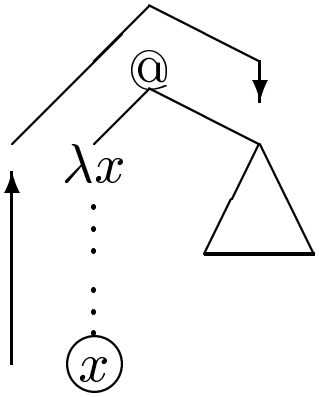
$$\begin{aligned} S &= F@\mathtt{a} \\ F &= \lambda_x(\mathtt{f}@v_x@(F@(\mathtt{g}@v_x))) \end{aligned}$$



The trees we consider are <u>not</u> regular

– but their definition is!

Introducing β-Redexes



- We are interested in automata runs over trees
- But our grammar is a definition of functions
  (Naturally given by λ-terms.)

$\rightsquigarrow$ when hitting a variable go and look for its argument!

$\therefore$ can work with a lambda tree – and don't have to unfold the grammar

At least for redexes of ground types...

. . . and more β-redexes

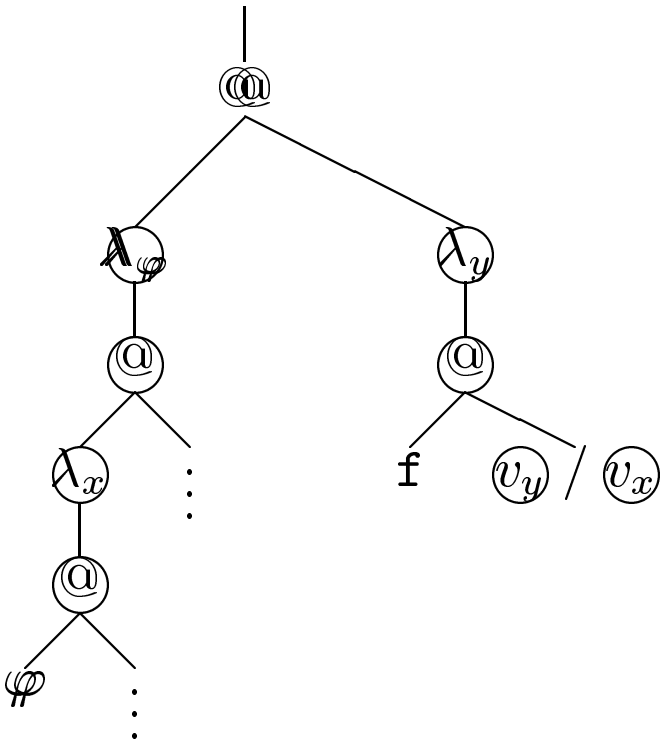
- MSO-define the relation "variable – denoted tree"

  or guess the traces of an automaton following these paths. . .

- But we have function abstraction as well!

$$N\overrightarrow{\varphi}\overrightarrow{x} = e \qquad \rightsquigarrow \qquad N = \lambda\overrightarrow{\varphi}\lambda\overrightarrow{x}e$$

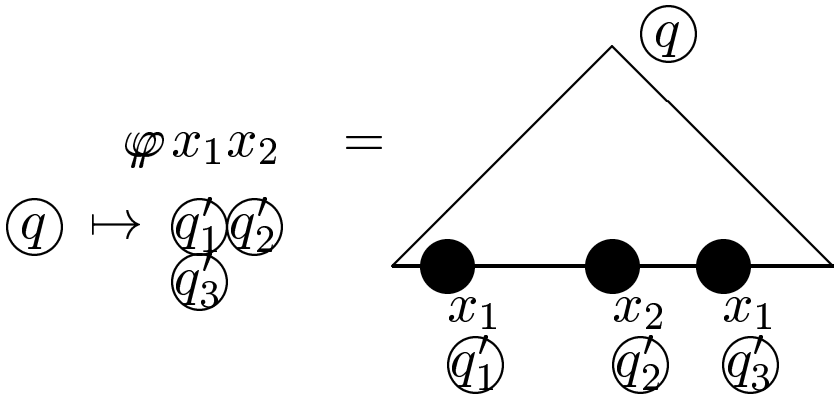
- Moreover: name capture. . .



How to describe a function variable φ?

- A function $\varphi\colon \iota \to \ldots \to \iota \to \iota$ is a tree with "holes" for the arguments
- For a run we only need to know with which states we continue at the various arguments.

! Note that we can identify several occurrences of the same argument



$\therefore$ finite description of φ.

Simulating a Run

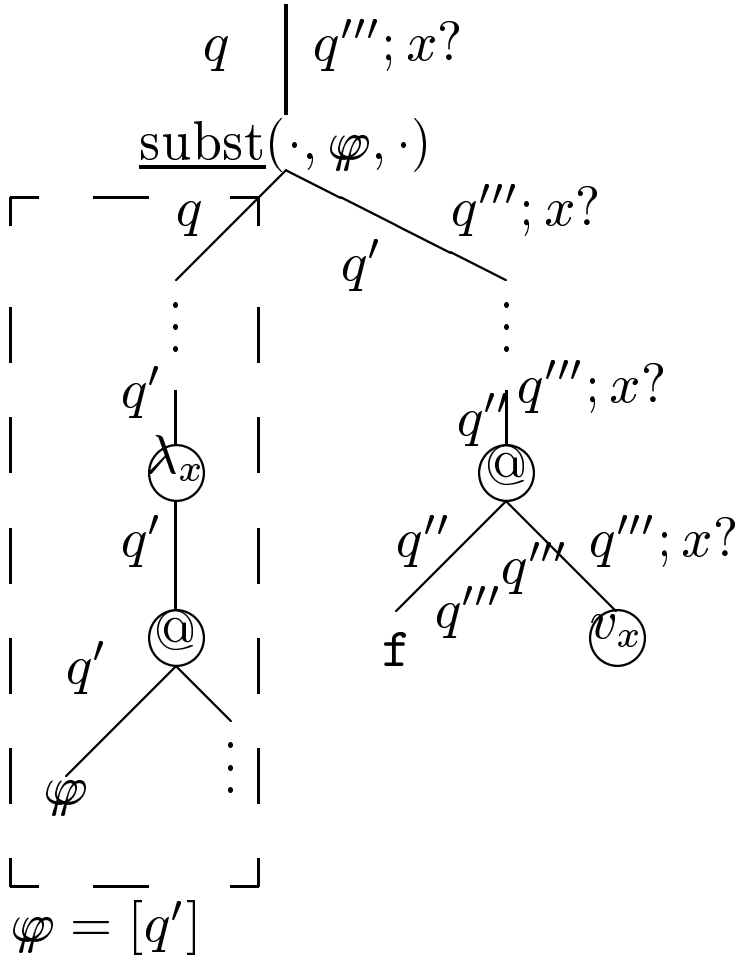
- All non-terminals take at most function arguments; right hand side of ground type

$\therefore$ all function arguments present

- write $\underline{\text{subst}}(t, \overrightarrow{\varphi}, \overrightarrow{t})$ for $(\boldsymbol{\lambda}\overrightarrow{\varphi}.t)\,\overrightarrow{t}$
- At each $\underline{\text{subst}}(\cdot, \overrightarrow{\varphi}, \cdot)$ guess the transition function for the φ_i

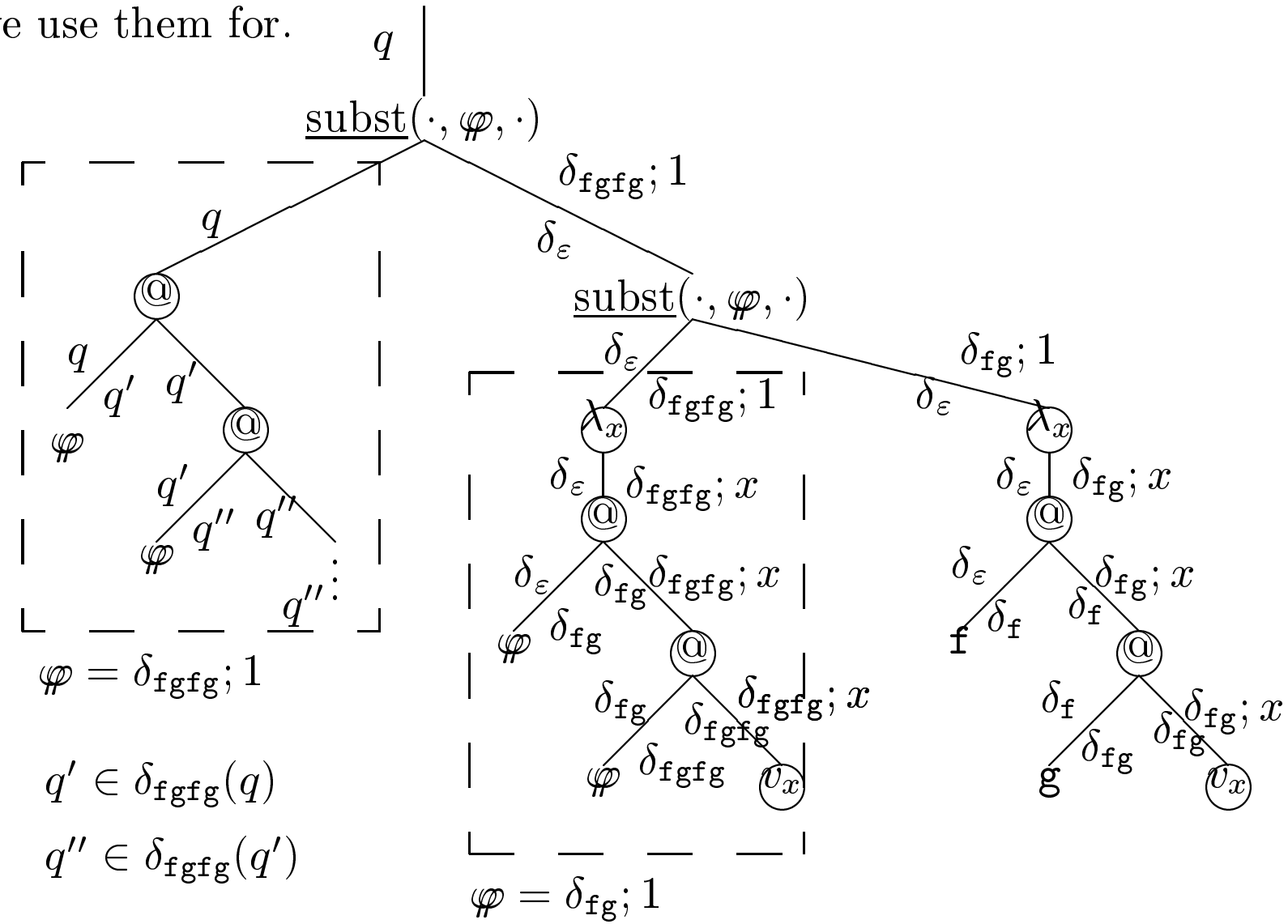
$\ldots$ In the main branch simulate the run with the "knowledge" of the transitions in the φ_i

In the side branches verify the guess.



Guesses Within Guesses

No problem! The guesses are absolute ones. We don't have to care what we use them for.



Summary of Argument

- Recursion scheme $\rightarrow$ lambda tree with $\boldsymbol{\lambda}$, @@

  This tree is regular.

  $\rightsquigarrow$ described by some automaton

- MSO formula $\rightarrow$ automaton accepting trees with this property

  Simulation...

  $\rightarrow$ another automaton accepting trees that denote (after β-reductions) trees with the given property

- Intersect these two automata

  $\rightsquigarrow$ automaton accepting trees that come from the grammar and have the given property

- Check emptiness of the last automaton *(decidable!)*