From Tree-Based Generators to Delegation Networks

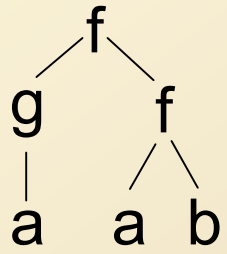
Frank Drewes
Umeå University

Structure

1. Tree-based generation
2. Types of picture generators covered (with examples in TREEBAG)
3. Delegation networks
4. Example
5. Some initial results
6. Future work

Tree-Based Generation (1)

- A tree (or term) is a formal expression over symbols from a ranked alphabet (or signature).

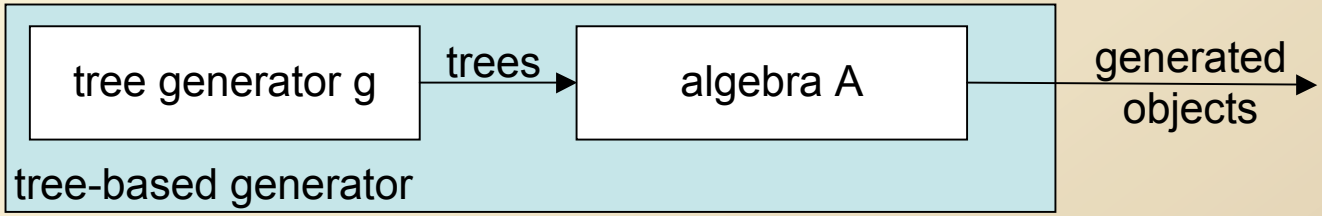


= f[g[a],f[a,b]]

- An algebra (or interpretation) associates every symbol with an operation on a chosen domain.
- Given an algebra, we can evaluate trees recursively.

Tree-Based Generation (2)

- The anatomy of a tree-based generator:



- Formally:
  - T_Σ = set of all trees over the ranked alphabet Σ
  - g = any device generating a tree language $L(g) \subseteq T_\Sigma$
  - A = Σ-algebra over some domain D
  - $G = (g,A)$ generates $L(G) = val_A(L(g))$
    $= \{ val_A(t) \mid t \in L(g) \}$

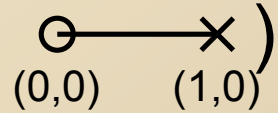
Tree-Based Generation (3)

What is it good for?

- Uniform framework for different types of devices
- Allows to exploit known results from the area of tree languages and tree transformations
- Variants of known devices are easily obtained:
  - syntactically, by changing the type of tree generator
  - semantically, by changing the type of algebra
- Easy to implement "incrementally"

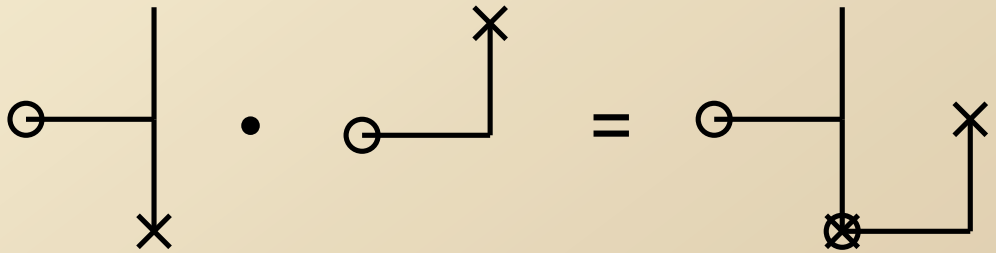
Types of Picture Generators Covered (1)

For example, chain-code picture grammars

- Picture = finite set of straight lines & end point
- Operations: constants l, r, u, d (i.e., r = 

)

  concatenation •

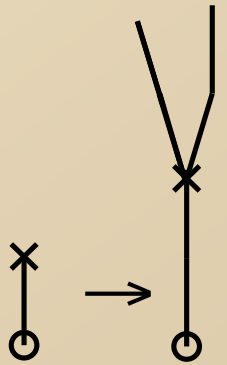


- Tree generator: regular tree grammar, ET0L tree grammar, etc.

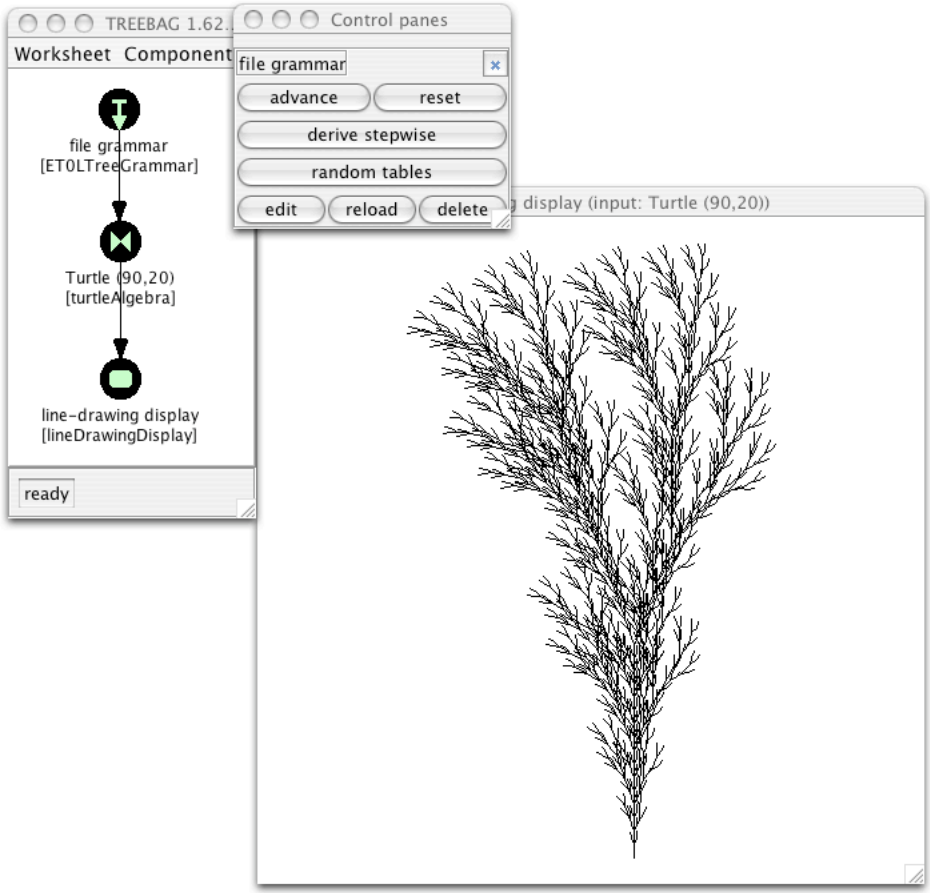
Types of Picture Generators Covered (2)

For example, L-systems with turtle geometry

- Operations: constant F (some line )
  rotations +, - by $\alpha°$, $-\alpha°$ (unary!)
  concatenation •
  enc (unary) sets end point to (0,0)
- Tree generator: ET0L tree grammar
- Example:
  F → F•F•enc[+[F•F]]•enc[-[F•+[F]]]]



Screen Shot (Example 1)



Types of Picture Generators Covered (3)

For example, collage grammars (slightly simplified)

- Picture = subset p of $\mathbf{R}^2$ (or $\mathbf{R}^d$)
- Operations: $F = \langle \alpha_1 \ldots \alpha_k, p \rangle$ of arity k

affine transformations — picture

$$F(p_1,\ldots,p_k) = p \cup \bigcup_{i=1,\ldots,k} \alpha_i(p_i)$$

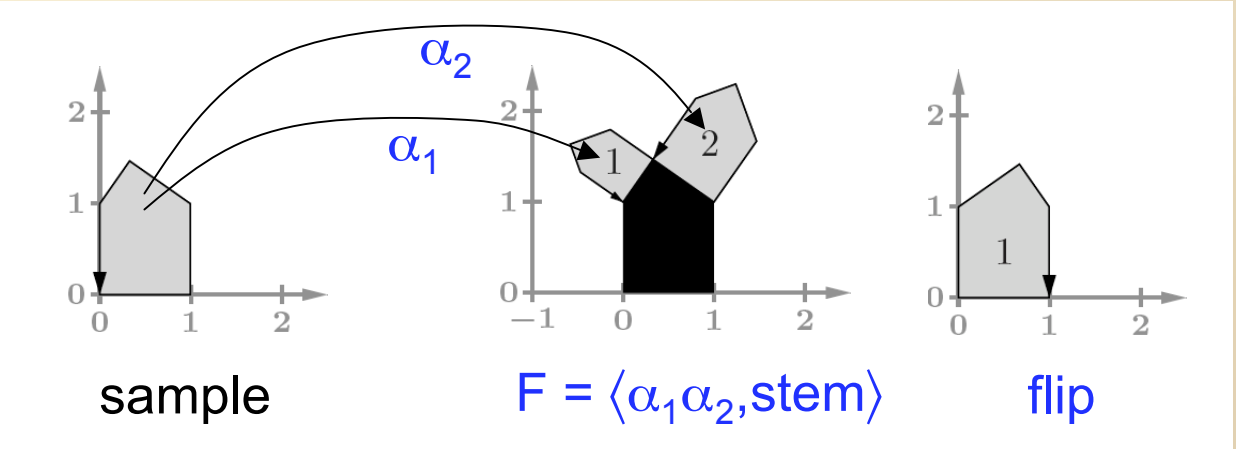
- Tree generators = regular tree grammar, ET0L…
- Example: …

Types of Picture Generators Covered (4)

A context-free collage grammar:

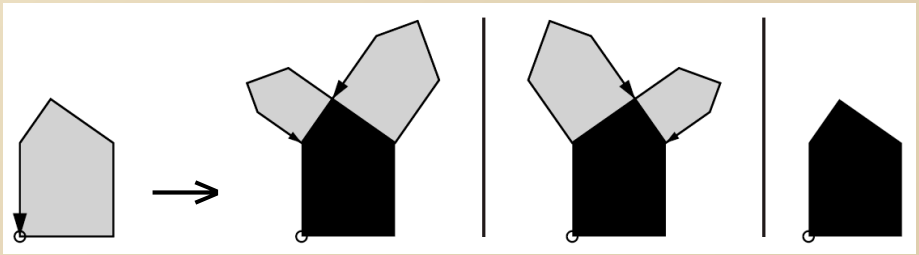
Operations:

- F (binary)
- flip (unary)
- stem (nullary)

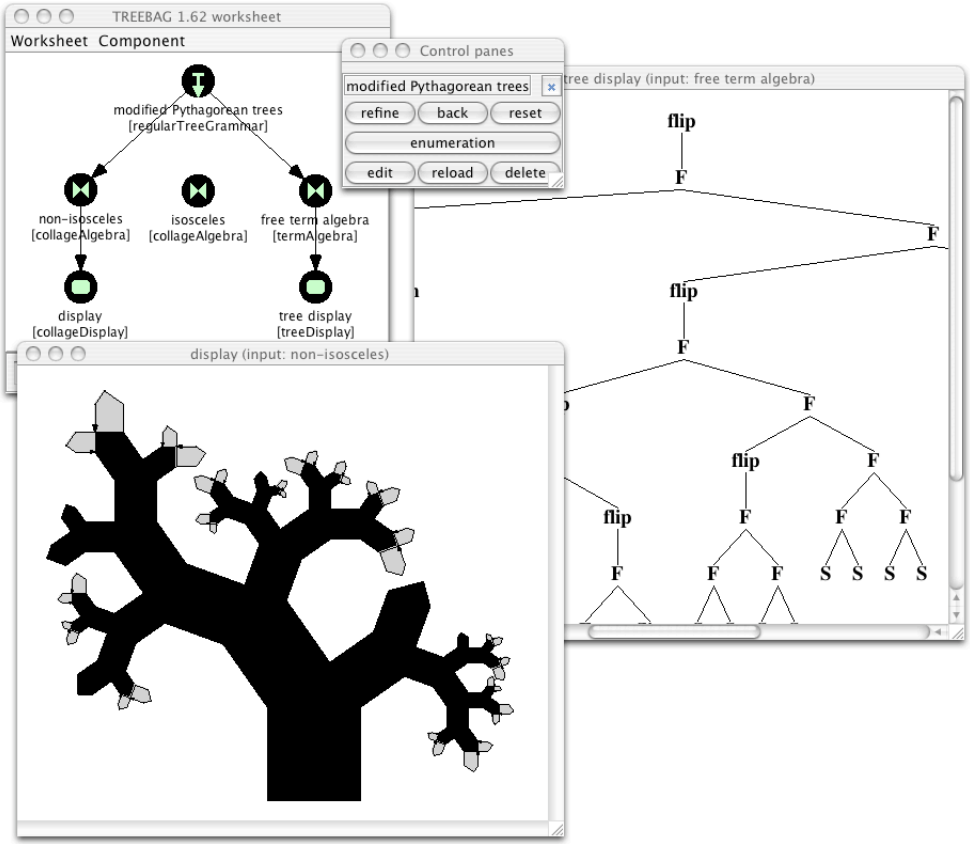


Rules (regular tree grammar):

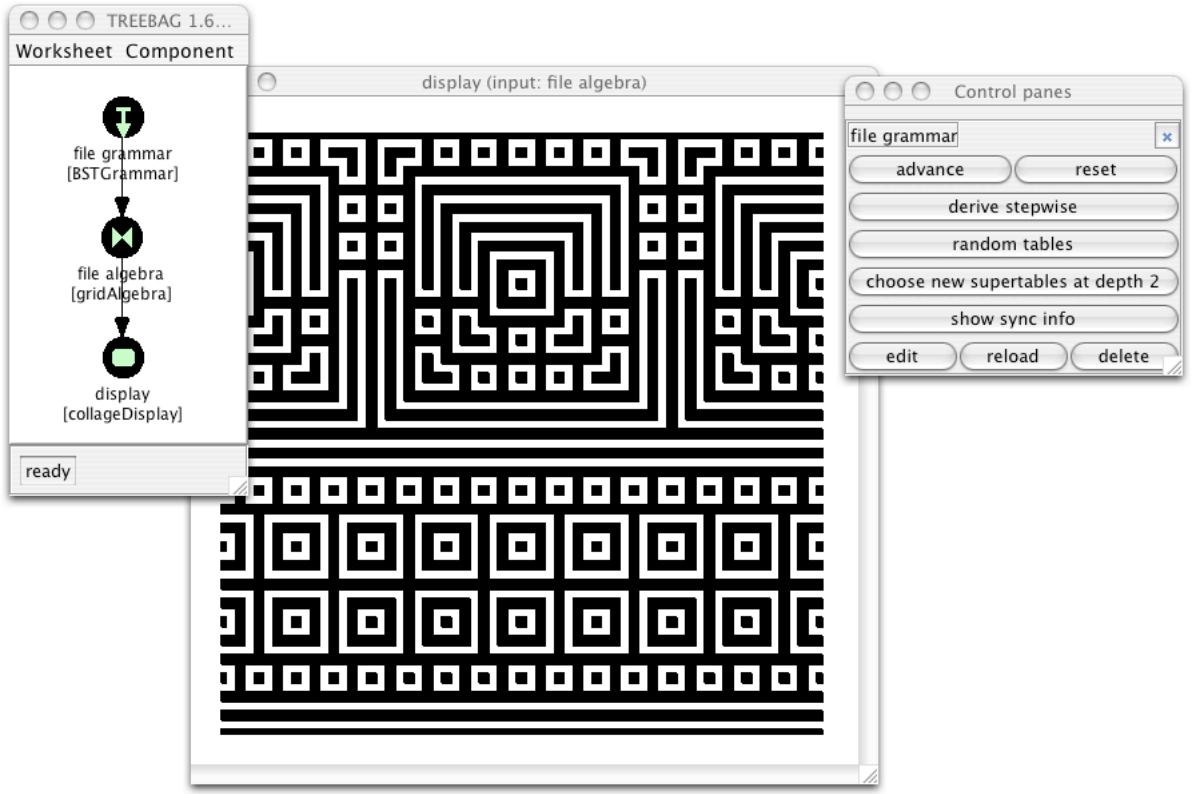
- S → F[S, S],
- S → flip[F[S,S]],
- S → stem



Screen Shot (Example 2)



Screen Shot (Example 3)



Delegation Networks (1)

Goals & motivation:

- Make tree-based (picture) generation suitable for practical applications (e.g., virtual realities).
- Build large generators in a modular way.
- Combine different types of tree-based generators.
- Allow to integrate other approaches into the tree-based setting (e.g., cellular automata).

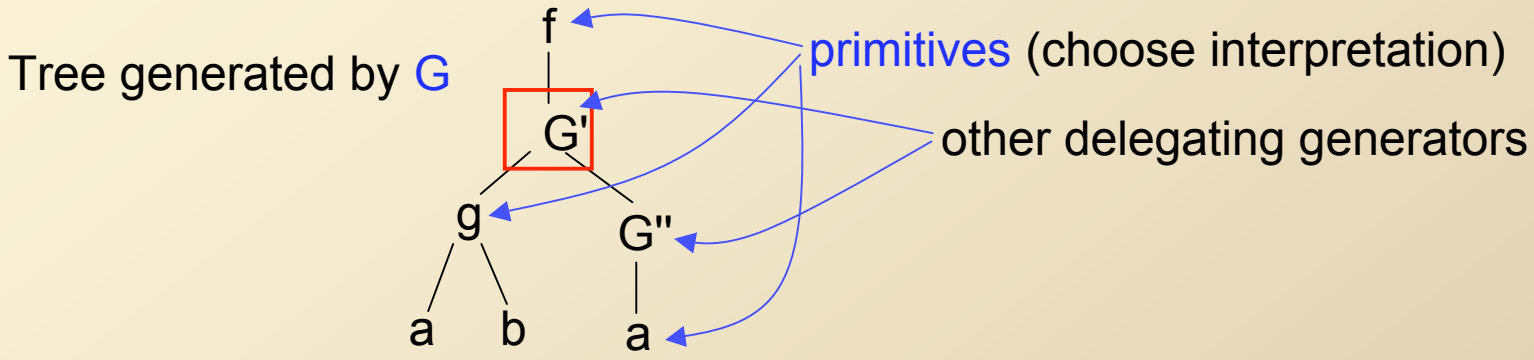
Delegation Networks (2)

The basic idea:

- Use a finite set **N** of delegating generators.
- $G \in \mathbf{N}$ is basically a tree-based generator, but…
- G has a rank, and can thus be used as a symbol.
- Its tree generator generates trees over 2 types of symbols:
  - primitives (interpreted as usual)
  - $G' \in \mathbf{N}$ (“calling” G')
- The second case is delegation.

Delegation Networks (3)

As a picture:



- G' should generate a function of arity 2.
- For this, the trees generated "inside" G' contain 2 parameter symbols ("formal parameters")
- Note: this leads to nondeterministic functions.

Delegation Networks (4)

Turning intuition into formalism:

- We consider several domains, i.e., many-sorted signatures and algebras.
- Symbols are interpreted as nondeterministic functions $f: D_1 x \ldots x D_k \rightarrow \wp(D)$.
- Given a signature X of (nullary) parameter symbols of sorts $D_1,\ldots,D_k$, every tree $t \in T_{\Sigma \cup X}$ evaluates to a nondeterministic function $val_A^X(t): D_1 x \ldots x D_k \rightarrow \wp(D)$.

Delegation Networks (5)

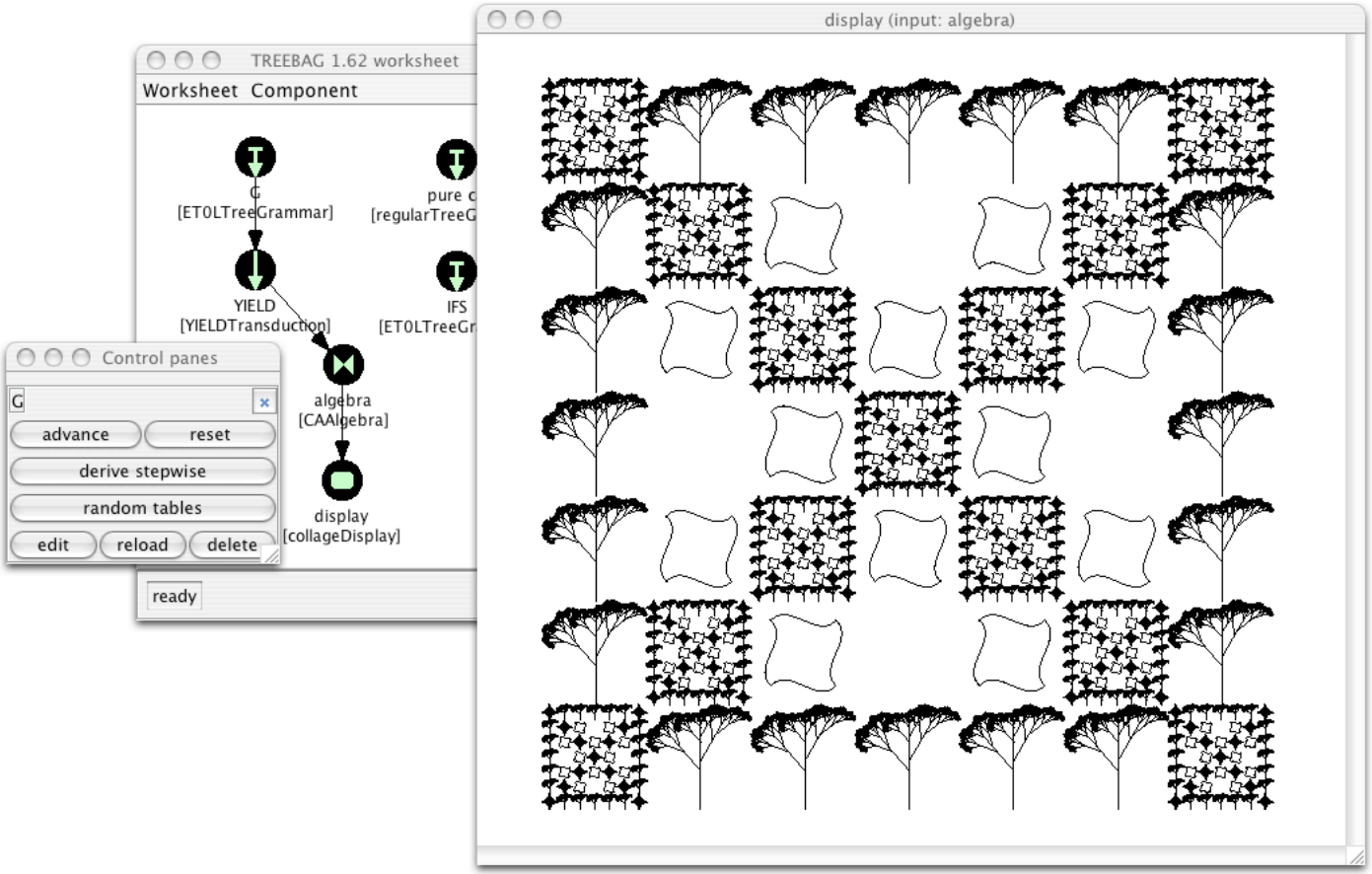
Semantics of a del. network N with primitives in $\prod$:

- Prerequisite: an interpretation $\pi(f)$ of all $f \in \prod$.
- We extend π inductively to interpretations $\pi_{N,i}$ of $\prod \cup N$. For every $G \in N$, let γ_G be its tree generator. Then
  - $\pi_{N,0}(G)$ = the nondeterministic function that yields no result at all
  - $\pi_{N,i+1}(G) = \mathrm{val}^{X}_{\pi_{N,i}}(L(\gamma_G))$. X
- Finally, $\pi_N(G) = \bigcup_{i=1,2,\ldots} \pi_{N,i}(G)$ for every $G \in N$.

Example

- Domains $\mathbb{N}$ (natural numbers), $\mathbb{P}$ (pictures in $\mathbf{R}^2$)
- Primitives: 0 and s on $\mathbb{N}$, collage operations, and cellular automata interpreted as ca: $\mathbb{N} \times \mathbb{P}^k \rightarrow \mathbb{P}$
- Delegating generators
  - IFS with $L(\gamma_{IFS}) = \{x, IFS[F[x,x,x]]\}$
  - CA with rules C → ca[n,C,y,z] | ca[n,x,y,z]
    generating all trees ca[n,ca[…ca[n,x,y,z],…],y,z]
  - G with rules S → CA[A,✦,IFS[⧫], ⬠]
    A → s[A] | 0

Screen Shot (Example 4)



Some Initial Results

- Using a suitable kind of IO-substitution, we can "flatten" the hierarchy of "trees within trees", thus defining the tree language $T_N(G) \in T_\Pi$ generated by a delegating generator.
- The interpretation of $T_N(G)$ w.r.t. π yields $\pi_N(G)$, (a "Mezei-Wright like" result).
- DEL(REGT) = YIELD(REGT) = DEL(FIN).
- Interestingly, $DEL^2(REGT) \not\subset YIELD^2(REGT)$.

Future Work

- Theoretical properties of delegation networks, e.g., characterize the "delegation hierarchy" $DEL^n(REGT)$, $n \geq 0$.
- Techniques to increase their efficiency
- Parallel and distributed execution
- Dynamic execution
- An implementation (similar to TREEBAG, but focusing more on efficiency and usability for "realistic" examples).

Thank you very much!

Questions? Comments?