# A Regular Query for Context-Sensitive Relations

Uwe Mönnich, Frank Morawietz, and Stephan Kepser

{um,frank,kepser}@sfs.uni-tuebingen.de

http://tcl.sfs.uni-tuebingen.de

Seminar für Sprachwissenschaft

Theoretical Computational Linguistics Group

SFB 441: Linguistic Data Structures

University of Tübingen



# Fundamental Problem

Skylla (Lack of Expressive Power)

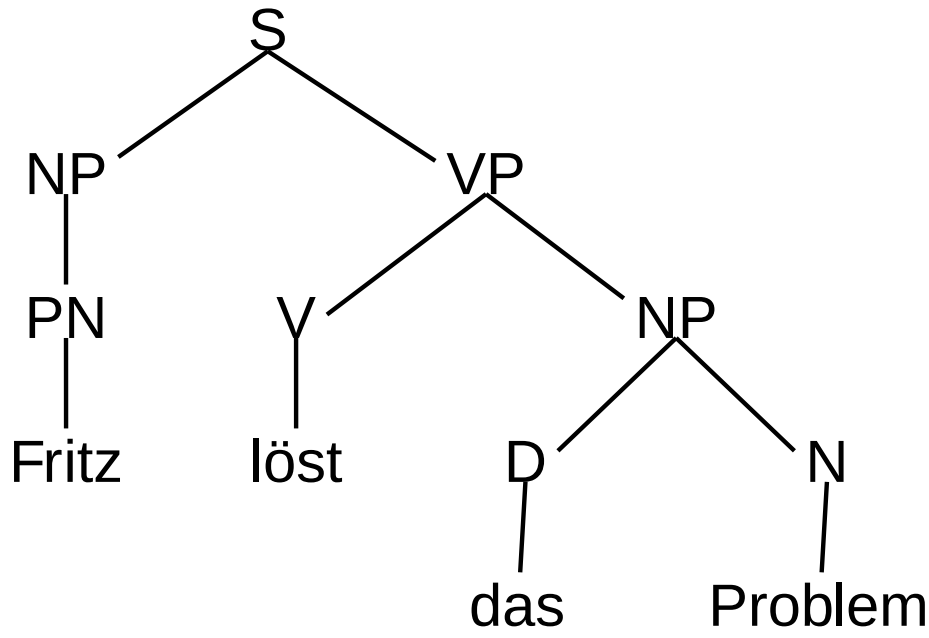
and

Charybdis (Undecidability)

# Core XML

(Dan Suciu: “The longest definition (500 pp) of regular tree grammars”.)

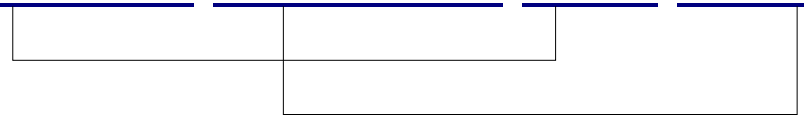
```
<S>
   <NP>
      <PN> Fritz </PN>
   </NP>
   <VP>
      <V> löst </V>
      <NP>
         <D> das </D>
         <N> Problem </N>
      </NP>
   </VP>
</S>
```



# Context-Sensitive Relations

Cross-Serial Dependencies:

*(. . . wil) mer de maa em chind lönd hälffe schwüme*



Swiss-German: $a^n b^m c^n d^m$—Non-CF

Cannot be queried by regular means.

Monadic Second Order logic (MSO) as query language is not powerful enough.



# Context-free Tree Grammars

- Needed to describe context-sensitive relations.
- Generalizations of context-free (string) grammars.
- Rules have the form $F(x_1, \ldots, x_n) \rightarrow t$
  $t$ is a tree with variables $x_1, \ldots, x_n$.
- Rewrite a non-terminal $F$ with complete tree $t$.
- Examples: TAG, Minimalist Grammars (E. Stabler).
- Not necessary to write a new grammar for every query.



# Query

Use MSO as query language.
A query is an MSO-sentence.
Example:

$$(NP_{akk} \cup NP_{dat})^* \cdot * \cdot (V_{akk} \cup V_{dat})^*$$

In praxi, use tree automaton representing the MSO-sentence.

Result: set of candidate trees,
all the trees we actually search for are in it,
but also a lot of garbage.



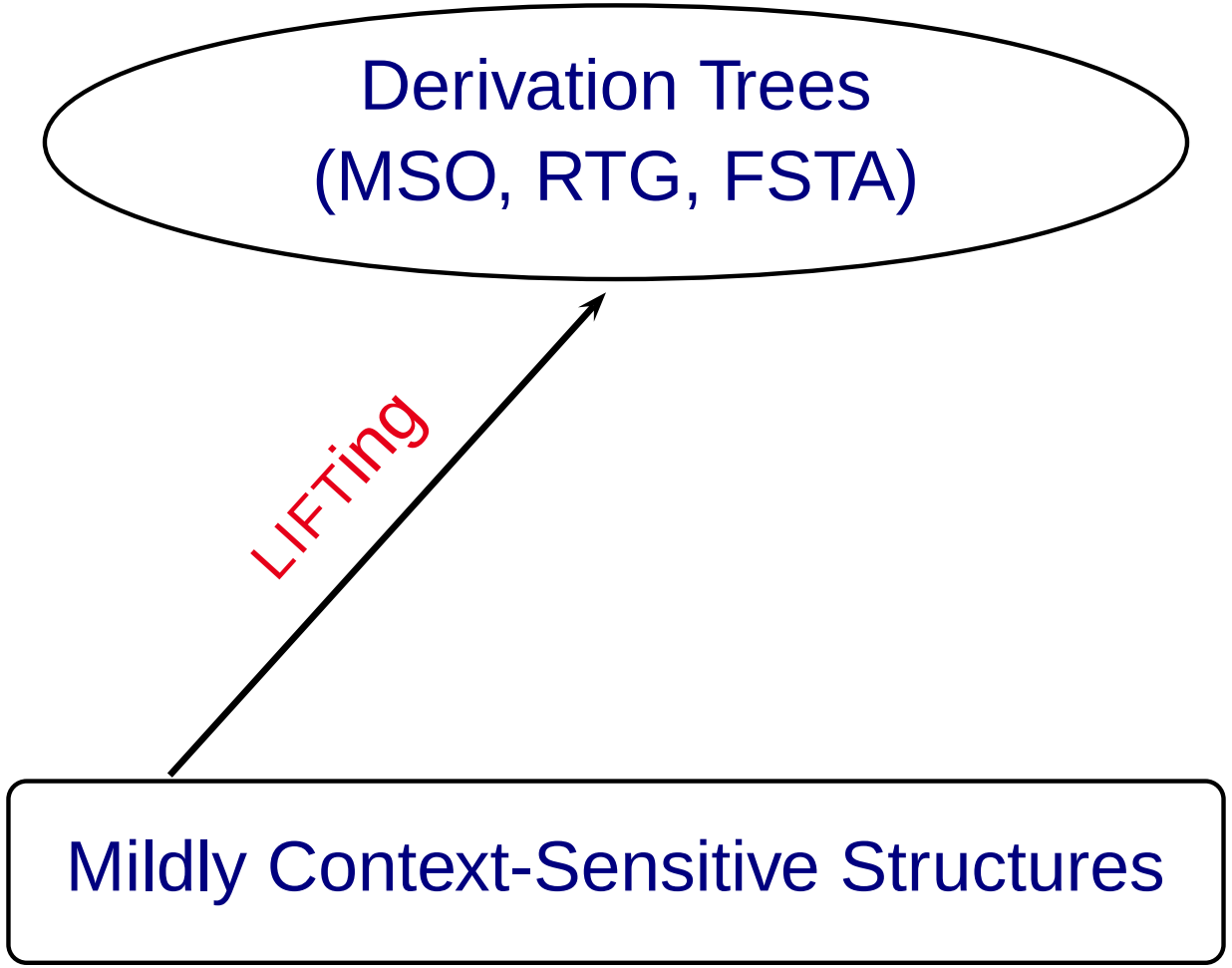
# Overview

Derivation Trees
(MSO, RTG, FSTA)

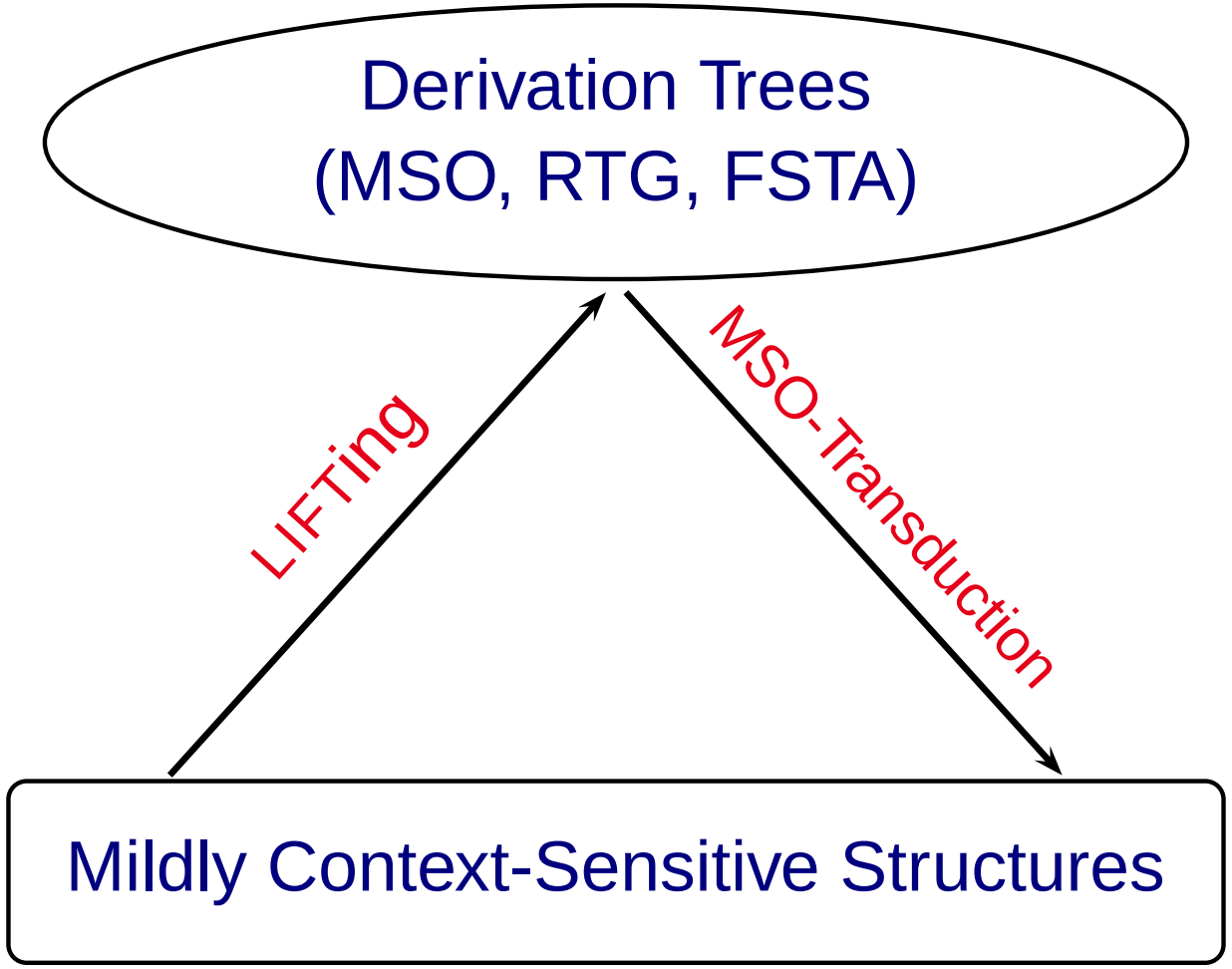
Mildly Context-Sensitive Structures



# Overview


Derivation Trees
(MSO, RTG, FSTA)
LIFTing
Mildly Context-Sensitive Structures

# Overview


Derivation Trees
(MSO, RTG, FSTA)
LIFTing
MSO-Transduction
Mildly Context-Sensitive Structures



# Lifting the Grammar

- Is a simple primitive recursive function
- Makes control structure visible
- All function symbols become constants.
- Resulting grammar is *regular*.
- Therfore expressible by an MSO-formula, and
- There is a tree automaton for the lifted grammar.

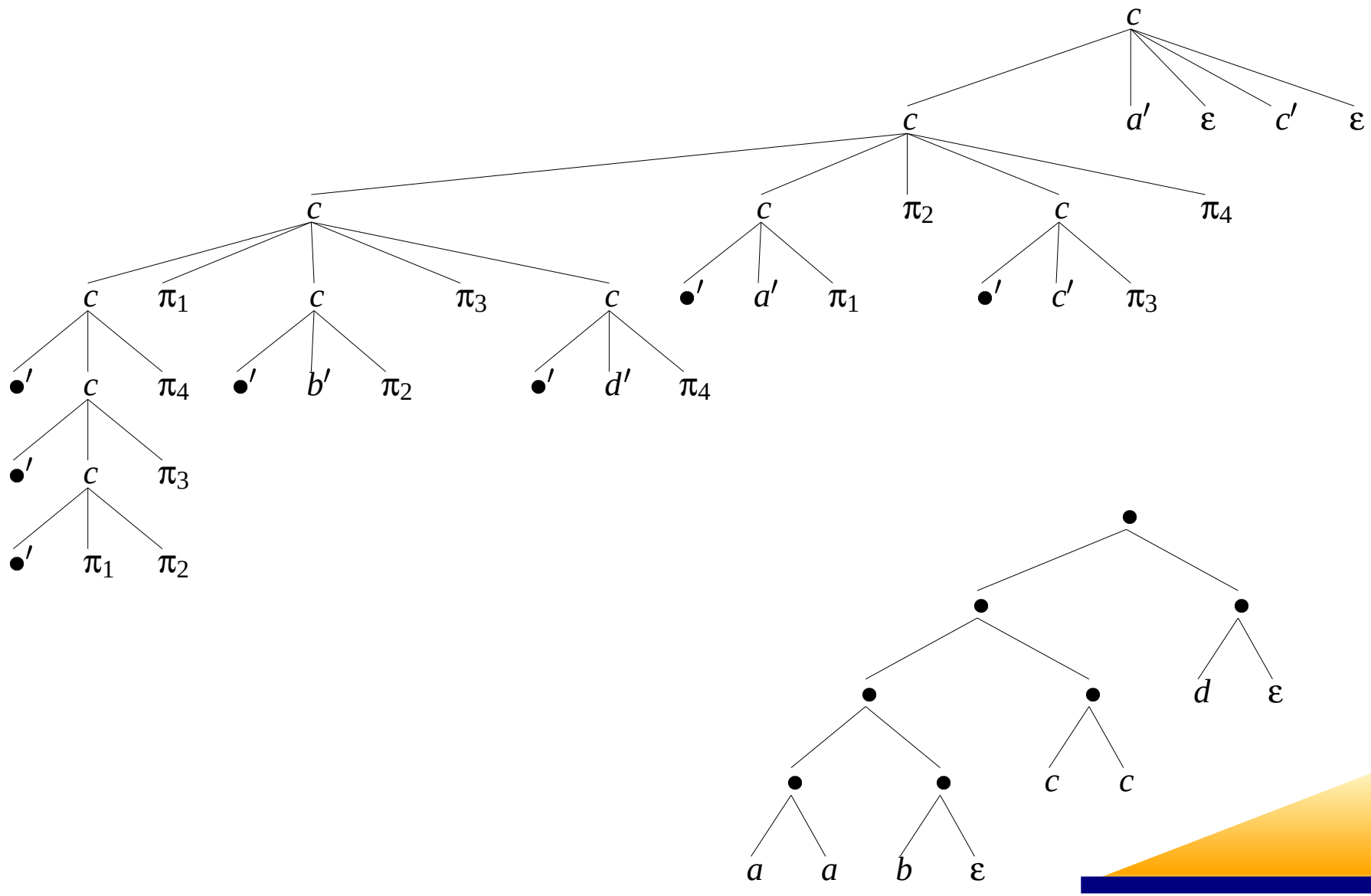


# Lifting the Candidate trees

- Similar to lifting a grammar.
- All internal nodes become leaves.
- For each candidate tree,
  there is a small finite set of lifted trees.



# Example of a Lifted Tree



# Lifted Query

We have

- a tree automaton representing the lifted grammar,
- and a set of lifted candidate trees.

Run the tree automaton on the set of lifted candidate trees.

Result: Solution set of lifted trees.

But: Lifted Trees are unreadable, not in the format of trees in the treebank.

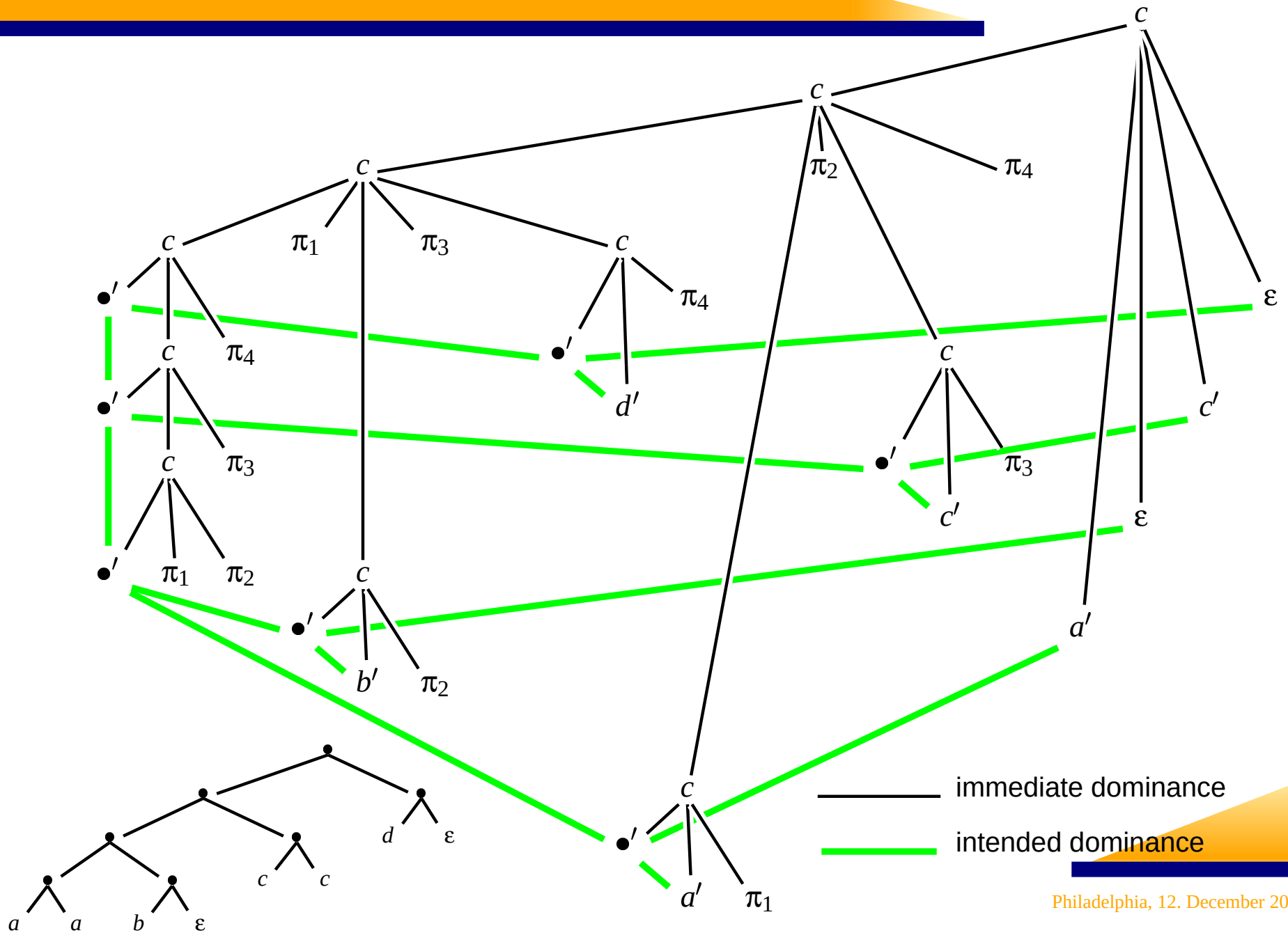


# Reconstruction of the Intended Structures

- The intended tree is present in the lifted tree, but hidden.
- Have to read the intended tree off the lifted tree.
- Technically: Define dominance and precedence relation of the intended tree
  on the basis of the dominance and precedence and the control structure (composition and projection) of the lifted tree.
- Possible with MSO-definable Transductions



# Reconstruction of the Intended Structures



# Intended Query Result

- Context-free tree grammar for describing context-sensitive relations
- MSO query on the original tree bank
  ➪ set of candidate trees
- Lift grammar and candidate trees
  ➪ automaton and set of lifted candidate trees
- Run automaton representing lifted grammar on lifted candidate trees
  ➪ set of lifted solution trees
- Reconstruct intended trees via MSO-transduction
  ➪ set of intended solution trees



# Effectiveness

- MSO decidable on trees
- MSO undecidable on graphs
- MSO + one binary relation undecidable on trees
- MSO → tree automata hyperexponential
- “Normalized” MSO → tree automata exponential
- Lifting and transduction require linear time