On Semistructured Data and Lingustics

Frank Morawietz

(joint work with Uwe Mönnich and Stephan Kepser)

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

Seminar für Sprachwissenschaft
Theoretische Computerlinguistik
Universität Tübingen

15. April 2002

Overview

- Introduction and Motivation
- Semistructured Data
- Query Languages
- Intermediate Conclusion
- Secondary Edges
- Regular Queries of Context-Sensitive Relations
- MSO Queries
- Lifting
- Reconstruction
- Conclusion and Outlook

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Introduction and Motivation

The Two Cultures

- exchange of information
- The Web
  - decomposition into units: files
  - structure (internal and between documents)
- Databases
  - relational database schemata or entity relationship diagrams
  - logical or abstract vs. physical implementation
  - concerns
    - query languages
    - mechanisms for concurrency control and for recovery
    - efficient implementation

Why do we have to bridge the gap?

- TV station A publishes data about political parties
  - stored: relational database
  - output is generated on demand in HTML via the formatting of the results of an SQL query
- Party B wants to analyze A's data
  - access only to A's HTML pages
  - write software for the analysis (parsing of HTML)
- Problems:
  - brittle
  - need to download many pages to get at the desired information

What we have . . .

- The Web
  - global infrastructure and standards for document exchange
  - presentation of hypertexts
  - user interfaces for document retrieval (IR techniques)
  - XML (data with structure)
- Databases
  - storage techniques and query languages with efficient methods for large amounts of (rigidly) structured data
  - data models and methods for structuring data
  - mechanisms for maintaining integrity and consistency
  - new model: semistructured data (relaxation of the demands on the structuring)

Basic Architecture

- Databases: client/server architecture
  - client issues a query
  - processing, optimization and execution
  - answer is returned by the server
- Web: “multitier” architecture
  - lowest level: data sources (database or file servers, legacy systems, “anything that produces data”) in a common logical data model and a common format (XML)
  - highest level: user interfaces or analysis software
  - middleware: transforms, integrates, adds data (data warehousing, mediator systems)
  - queries: translation, processing, translation

Data Abstraction

- databases: three levels
  - physical
  - logical
  - external
- web
  - no distinction between physical and logical layer
  - is there a reason why we should make this distinction?
  - Example: database can be a single XML file, or a compressed version, or a relational database, or . . .

➪ XML as both physical and logical representation

Data Diversity

- Heterogeneous data
- How is the data conceptually represented?
- Example: addresses – sorted by name? sorted by town?
- Reconciliation at the logical level is possible

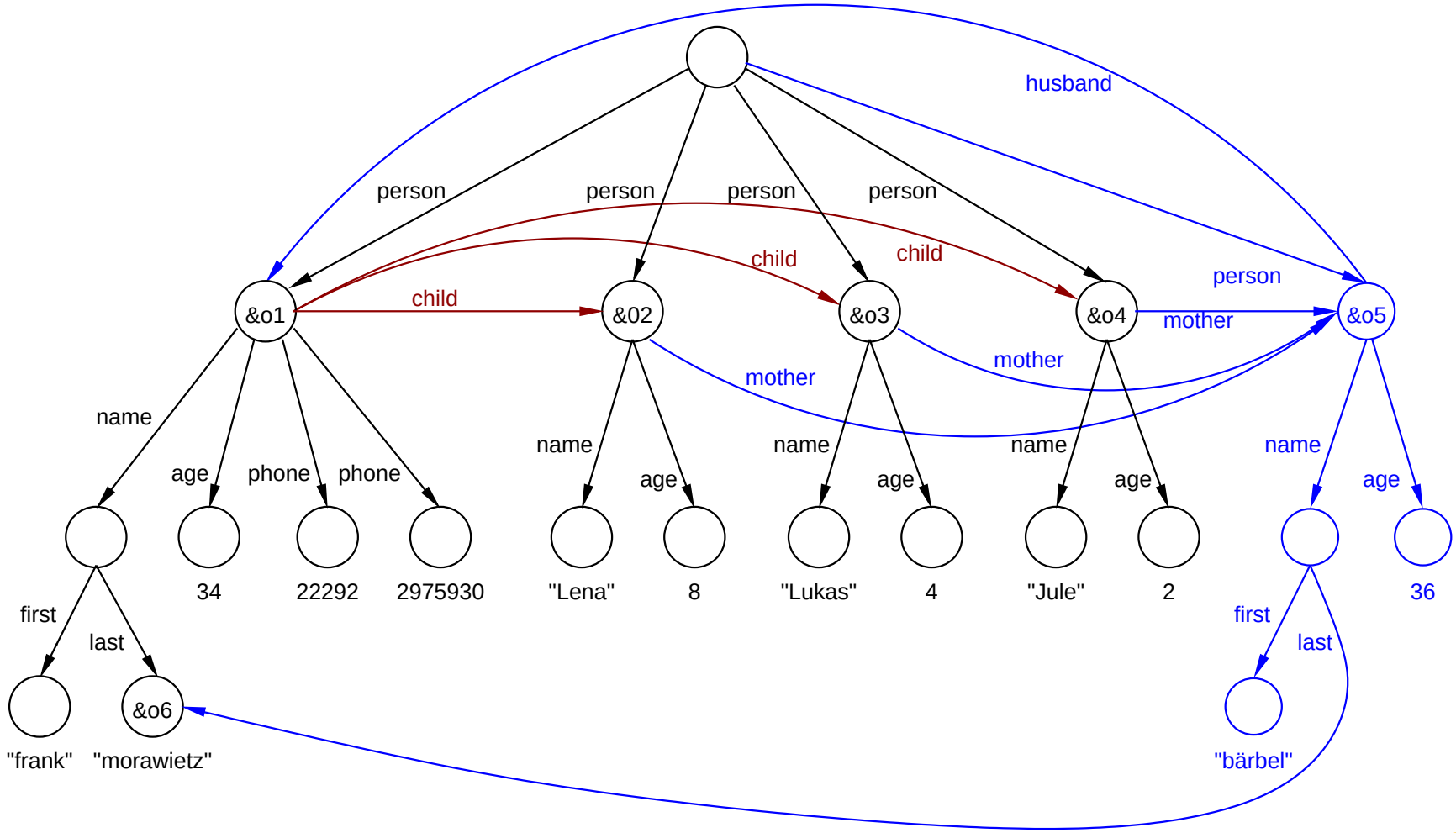
➪ XML for the logical level

Research Context

- **Semistructured Data**
  - incompleteness and irregularity of data
  - self-describing
  - integration of several kinds of data
  - no relational schemata, but structured
- **XML**
  - universal format for data exchange
  - data model: labeled graphs/trees
  - query languages

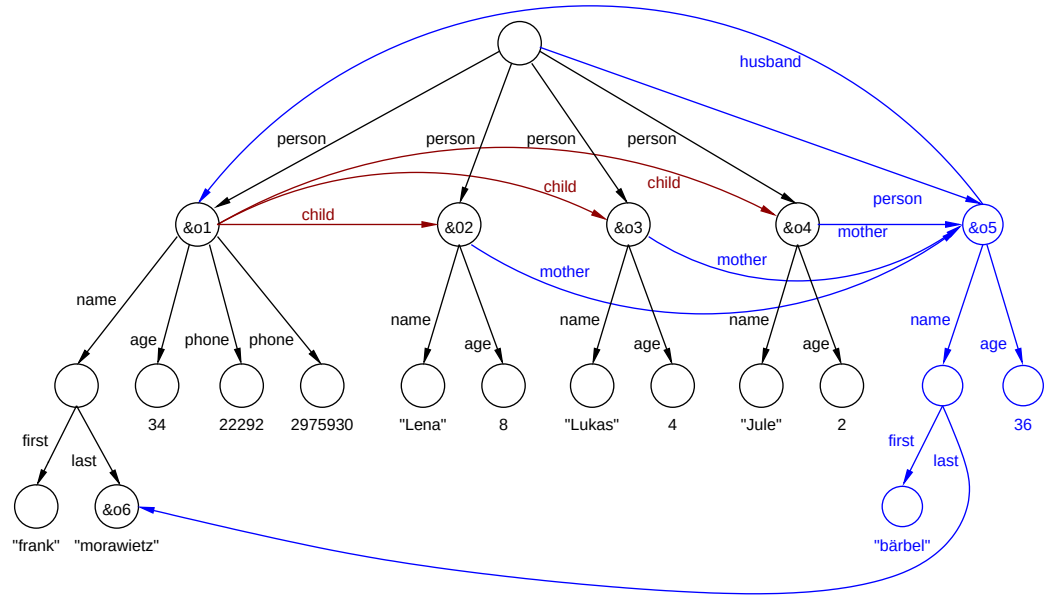
Semistructured Data

The Data Model: Syntax



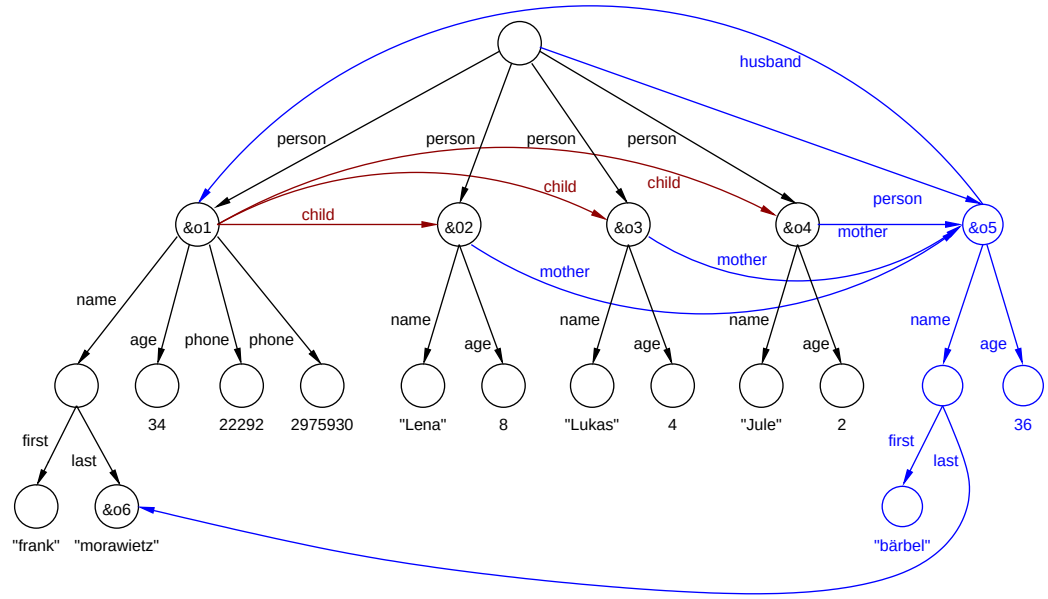
The Data Model: Syntax

```
{name: "Lena", age: 8}
```



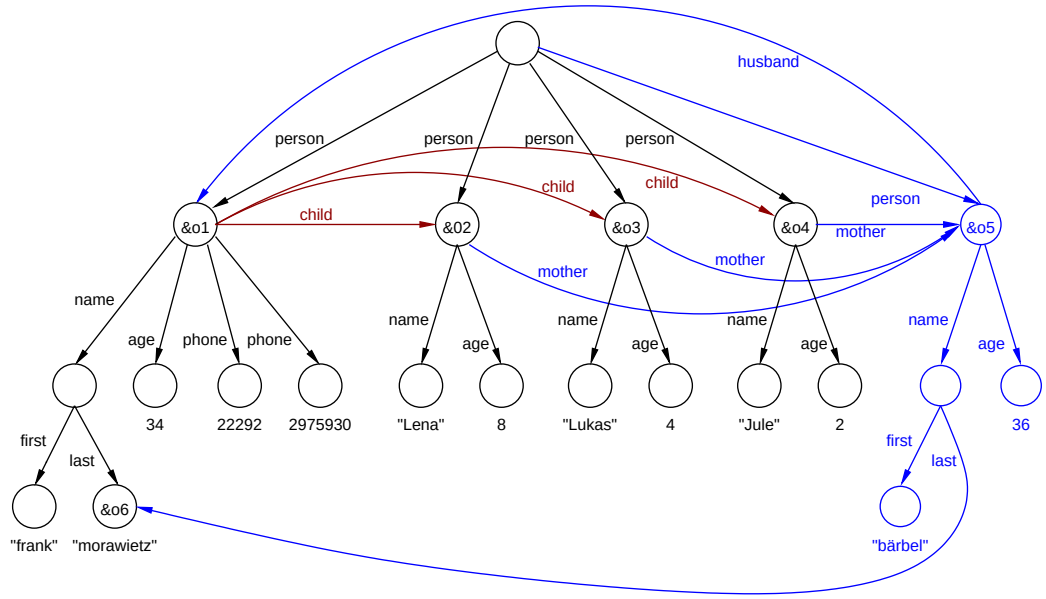
The Data Model: Syntax

```
{name: {first: "frank", last: "morawietz"},
age: 34,
phone: 22292}
```



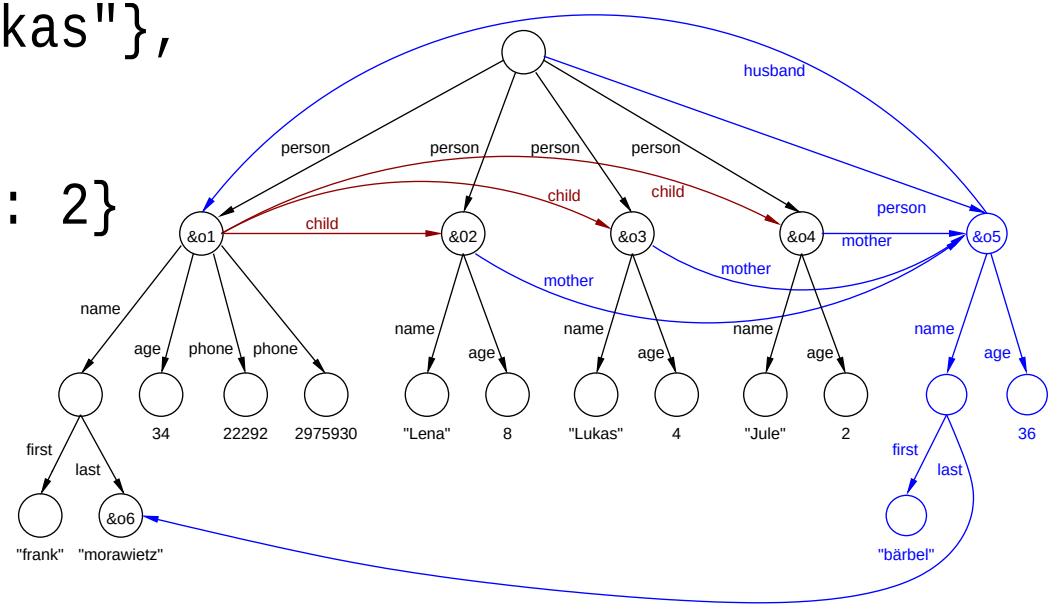
The Data Model: Syntax

```
{name: {first: "frank", last: "morawietz"},
age: 34,
phone: 22292,
phone: 2975930}
```



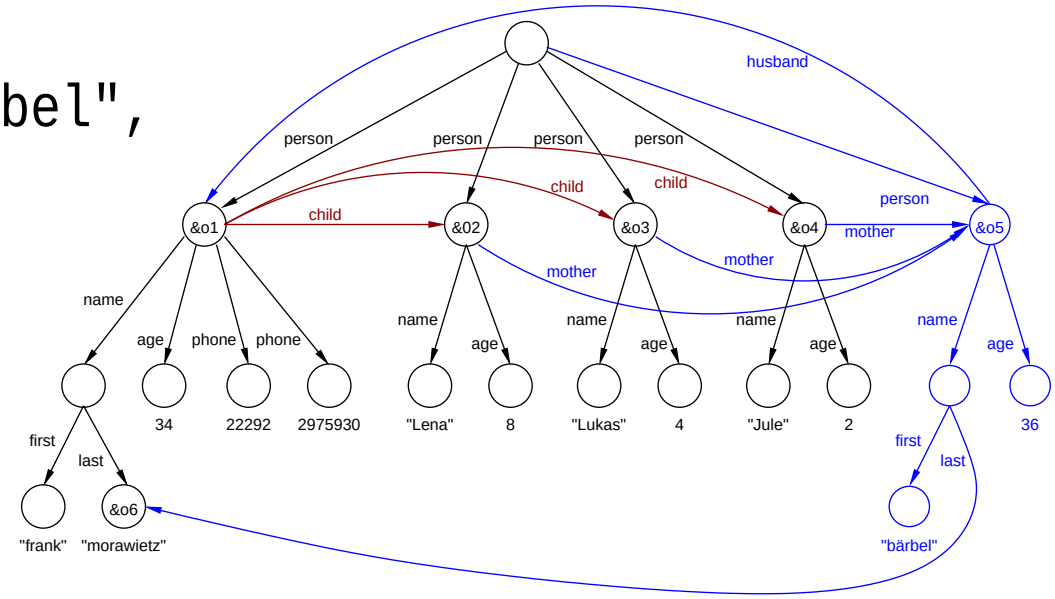
The Data Model: Syntax

```
{
person: {name: "Lena", age: 8},

person: {age: 4, name: "Lukas"},

person: {name: "Jule", age: 2}
}
```



The Data Model: Syntax

```
{
person:
    &o2{name: "Lena", age: 8, mother: &o5 },
person:
    &o5{name: {first: "Bärbel",
               last: &o6},
       age: 36,
       husband: &o1
      },

...
}
```



The Data Model: Syntax

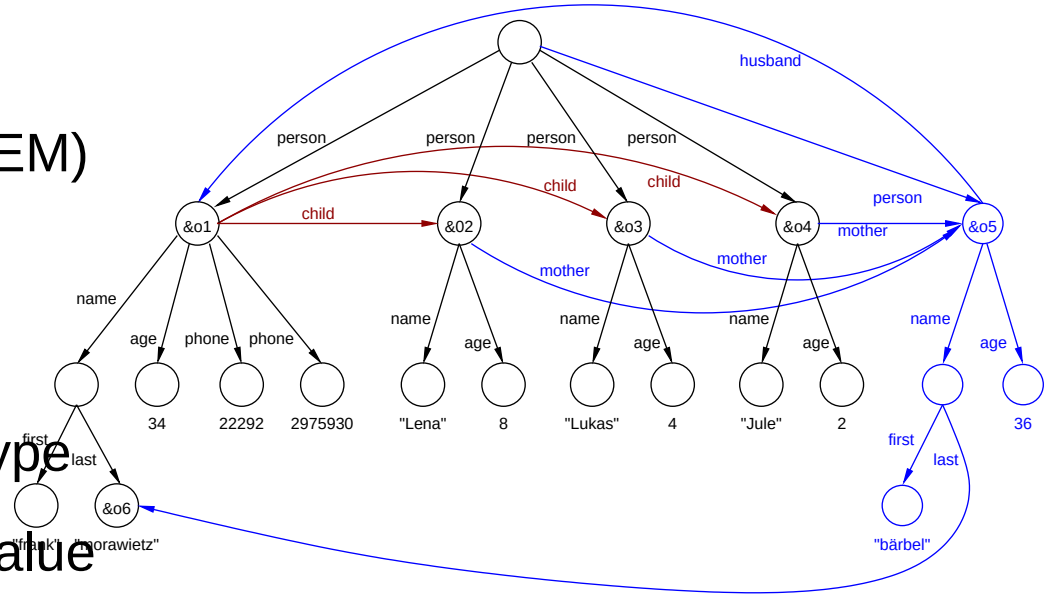
- Convention: implicit OIDs
- `{a: {b: 3}, a: {b: 3}}`
- coding of relational databases (e.g., via rows)
- edge labeled graph
- ⇨ Object Exchange Model (OEM)

label: string

oid: object identifier

type: complex or a specific type

value: list of OIDs or atomic value



Semistructured Data and XML

```
{ person: {name: {first: "frank", last: "morawietz"},
           age: 34,
           phone: 22292}}
```

```
<person>
  <name>
    <first> frank </first>
    <last> morawietz </last>
  </name>
  <age> 34 </age>
  <phone> 22292 </phone>
</person>
```

Semistructured Data in Linguistics

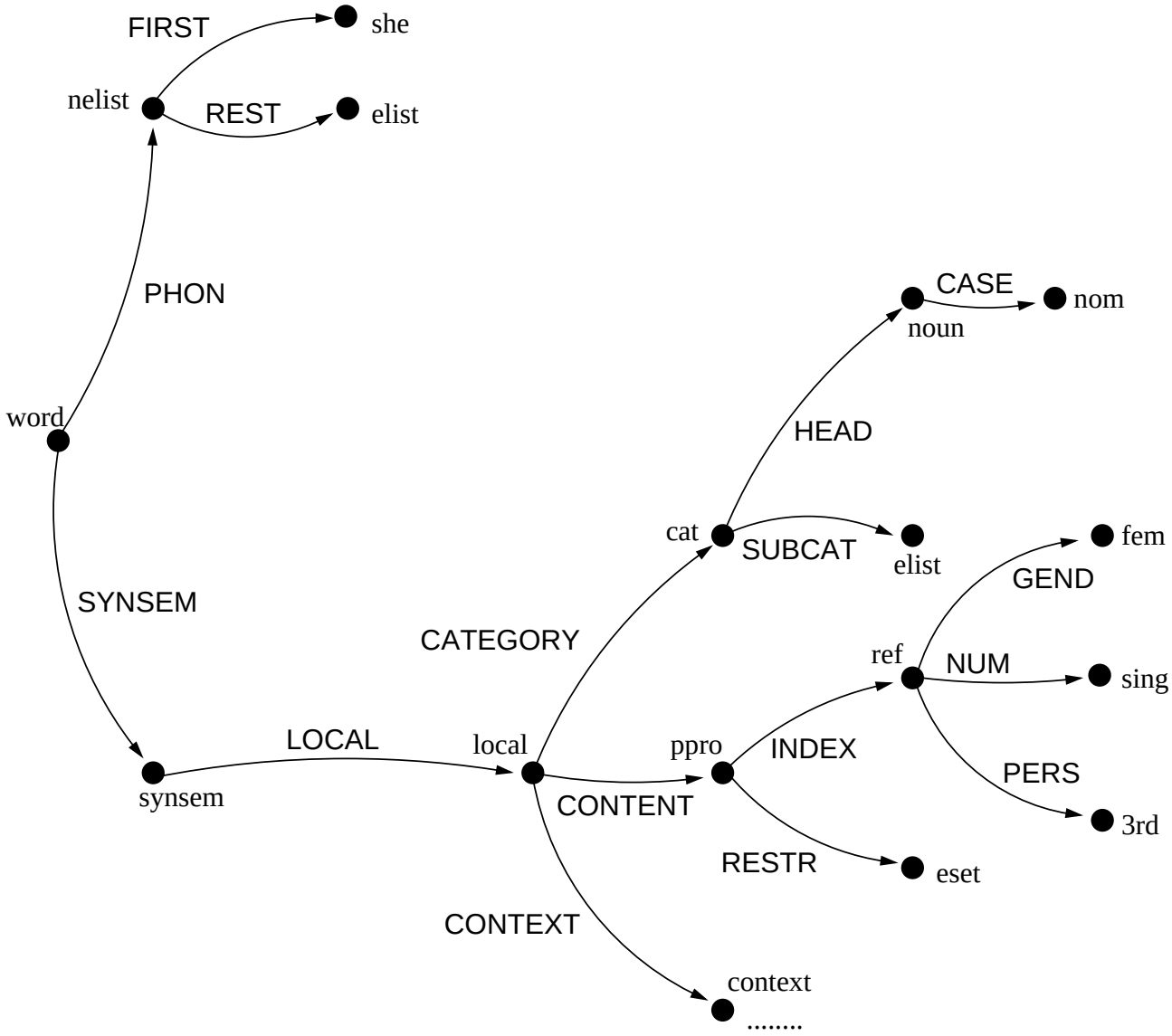
Linguistic Databases: collections of linguistic data

- relational databases (e.g., typological databases)
- lexicons
- corpora (e.g., books, newspaper articles, transcriptions, . . .)

Semistructured Data in Linguistics

Language	WO	Prep	Case	Adj	...
German	SOV	←	Acc	←	...
English	SVO	←	Acc	←	...
Warao	SVO?	→	Acc	→	...

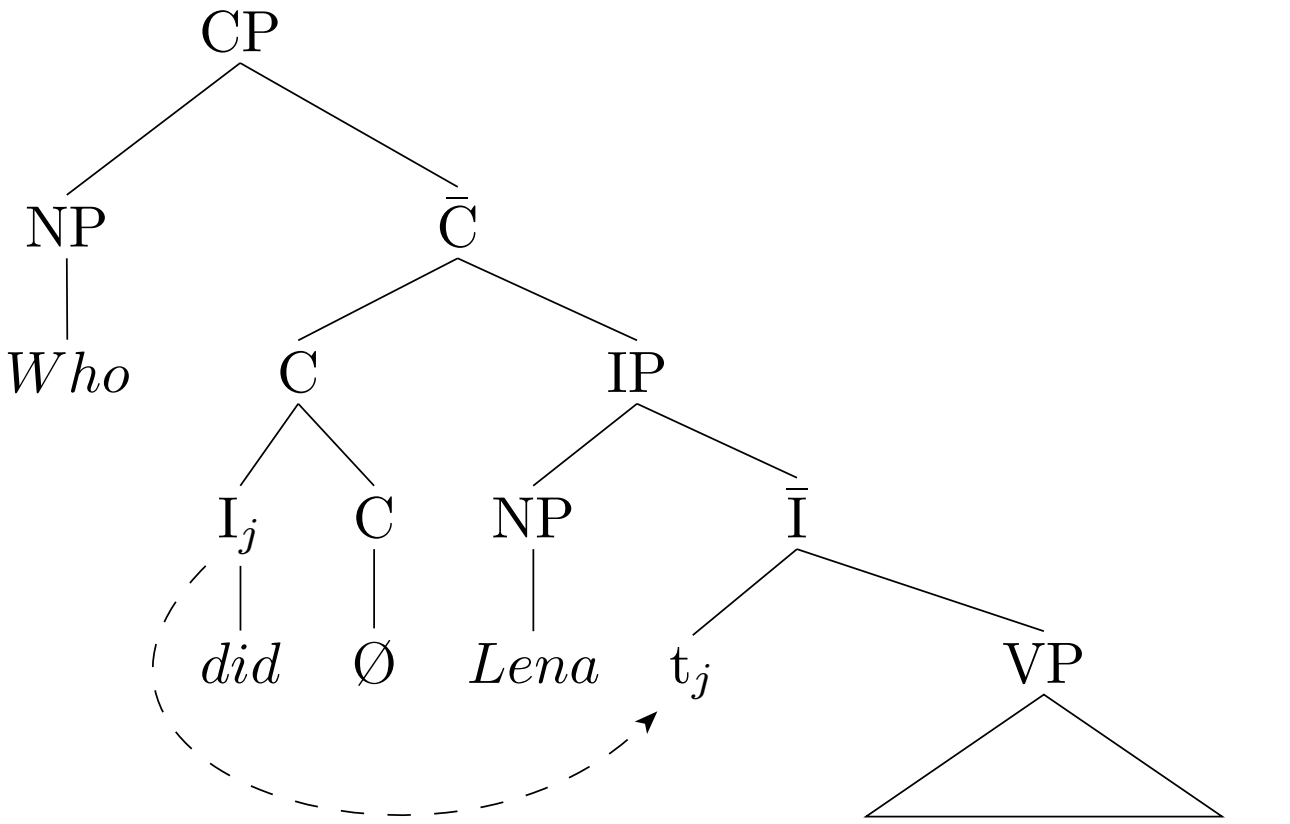
Semistructured Data in Linguistics



Semistructured Data in Linguistics

$$
\begin{bmatrix}
\textit{word} \\
\textit{phon} & \begin{bmatrix} \textit{nelist} \\ \textit{first}\ \textit{she} \\ \textit{rest}\ \textit{elist} \end{bmatrix} \\
\textit{synsem} & \begin{bmatrix} \textit{synsem} \\ \textit{local} & \begin{bmatrix} \textit{local} \\ \textit{category} & \begin{bmatrix} \textit{cat} \\ \textit{head} & \begin{bmatrix} \textit{noun} \\ \textit{case}\ \textit{nom} \end{bmatrix} \\ \textit{subcat}\ \textit{elist} \end{bmatrix} \\ \textit{content} & \begin{bmatrix} \textit{ppro} \\ \textit{index} & \begin{bmatrix} \textit{ref} \\ \textit{gend}\ \textit{fem} \\ \textit{num}\ \textit{sing} \\ \textit{pers}\ \textit{3rd} \end{bmatrix} \\ \textit{restr}\ \ \textit{eset} \end{bmatrix} \\ \textit{context} & \begin{bmatrix} \textit{context} \\ \ldots\ \ldots \end{bmatrix} \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

Semistructured Data in Linguistics



Who did Lena invite to her birthday party?

Semistructured Data in Linguistics

Penn Treebank II:

$(TOP\ (S\ (NP\text{-}SBJ$ my best friend$)$
$(VP$ gave
$(NP$ me$)$
$(NP$ chocolate$)$
$(NP\text{-}TMP$ yesterday$))$
$.))$

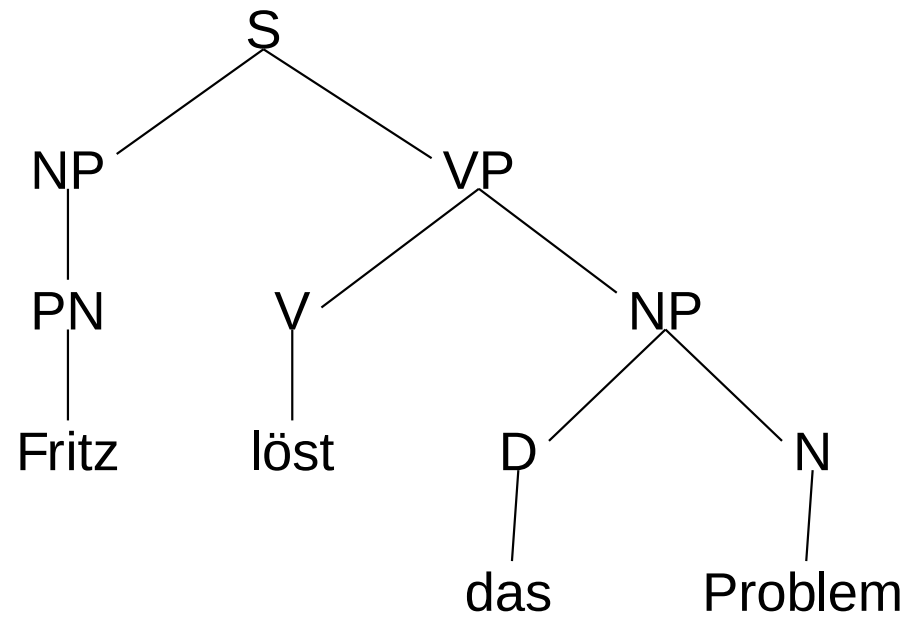
Annotation

Additional information to the “raw” data

- meta information
- document structure
- part-of-speech tags
- syntactical analysis
- specific syntactic or semantic phenomena
- …

Core XML

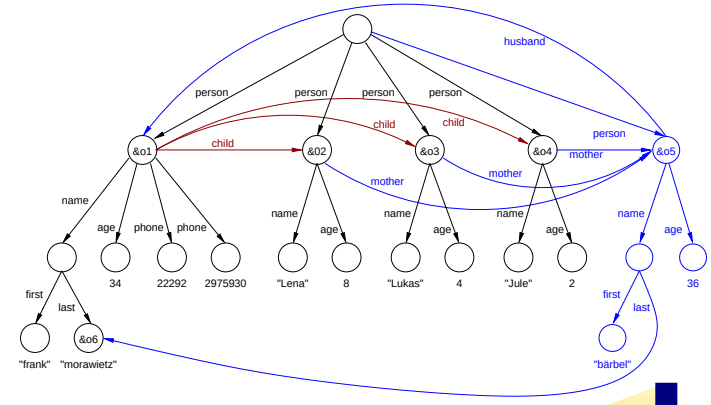
```
<S>
   <NP>
      <PN> Fritz </PN>
   </NP>
   <VP>
      <V> löst </V>
      <NP>
         <D> das </D>
         <N> Problem </N>
      </NP>
   </VP>
</S>
```



Query Languages

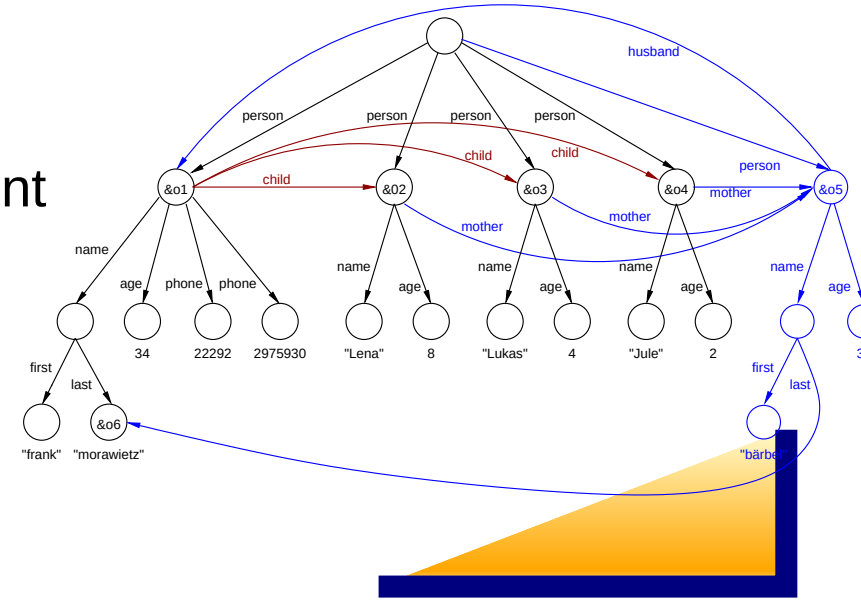
A simple Query Language

- path expressions (`"person.name.first"` yields the set of nodes with labels `"frank"` and `"bärbel"`)
- path expressions result in sets of nodes
- paths via properties: regular expressions (including a wild card)
  - on the alphabet of edge labels (`"person._*.name"`)
  - on the alphabet of the characters of edge labels (`(((n|N)ame)(s)?)`)
- Query syntax based on Lorel/UnQL
  - ```
    select firstname:  X
    from person.name.(first)?  X
    ```
  - ```
    select firstname:  X
    from person.name X
    where "frank" in X.first
    ```



More Complex Queries

- ```
  select person:  X
  from person X
       X.age Y
  where Y < 30
  ```
- ```
  select person:  {firstname:  Y, age:  Z}
  from person X
       X.name.(first)?  Y
       X.age Z
  ```
- nested queries in the `select` statement
- natural joins



Syntactic Sugar for Queries

- default labels
- path expressions in the `select` clause
- equality predicates
- coercion
- relation names as variables
- label and path variables

Desiderata for Query Languages

- Expressive Power
  - operations (comparable to, e.g., SQL)
  - restructuring
  - completeness
- Semantics
  - query transformation
  - query optimization
- Compositionality: output can be used as input of a query
  - same data model
  - referential transparency
- Structure Consciousness
- Program Manipulation

Reintroducing Structure: Why?

- Ignoring knowledge of types loses information
- Regularities facilitates querying.
- Protection against incorrect updates.
- Simplify access and improve performance.
- Optimization of the physical representation.
- Mixing querying structured and semistructured data.

- Maybe hidden from the user?

Query Languages for XML

- combine XML syntax with query languages presented previously
- XML-QL, XQuery, XSL(T)
- Problems:
  - Order
  - infinite domain of attributes

Advanced Topics

- first-order interpretation
- reconstruction
  - Datalog
  - Graphical Languages: GraphLog, Complete Answer Aggregates
  - Structural Recursion
  - Bisimulation
- Strudel/StruQL

TGrep (Penn Treebank)

`NP`	anything called just `NP`
`TOP`	anything called `TOP`. Since every sentence is enclosed in `TOP` brackets, this matches everything.
`VP < NP`	`VP` immediately dominates an `NP`
`VP <3 NP`	`VP` has `NP` as its 3rd child
`NP > VP`	an `NP` that has a `VP` parent
`VP << dog`	`VP` contains the word `dog` (`VP` dominates `dog`, directly or indirectly)
`S \!<< dog`	`S` does not contain the word `dog`
`NP $..  NP`	two `NP`s have a common parent (first `NP` precedes a sister also called `NP`)
`/^NP/`	`//` encloses a grep-style regular expression

TGrep (Penn Treebank)

```
S <1 /^NP/ < (VP < (NP $..  NP))
```

Get all Ss that start with an NP (not necessarily the subject) and that dominate a VP that in turn has two NP children – in other words, sentences with what might be double-object VPs.

TGrep2

- Rather than simply having a set of required relationships and a set of prohibited relationships, nodes can have full boolean expressions of relationships to other nodes.
- Nodes can be given unique labels and may then be referred to by those labels in the pattern specification or in selecting trees for printing.
- Patterns are no longer restricted to simple tree architectures. The use of node labels and segmented patterns allows links in a pattern to form back-edges as well, permitting cycles of links.
- Customizable output formats allow a variety of information to be reported in a flexible manner.
- Multiple search patterns may be specified and one can retrieve the first subtree matching any pattern, the first subtree matching each pattern, or all subtrees matching all patterns.
- Subtrees can be reported using a code rather than by printing the whole structure. The trees themselves can later be retrieved using the codes.
- A variety of new links have been added and the immediately-precedes link now has a more conventional meaning.
- TGrep2 corpus files are substantially smaller than tgrep corpora.

FXGrep (Neumann/Seidl)

- XML input as a document forest representing the logical structure
  - A text node represents one or more adjacent pieces of character data.
  - A PI node represents a processing instruction and is annotated with the target name of the processing instruction. It has a single child: a text node giving the text of the processing instruction.
  - An element node represents an element and is annotated with the element type and the attributes specified for the element. The children of an element node are a sequence of nodes constructed from the content of the element.
- query implemented by consecutive runs of two forest automata (for simple queries a single run suffices)

FXGrep: Pattern Language

`/doc/section/title`	paths
`/doc//title`	descendant
`//title`	descendant anywhere
`//(section/||subsection/)title`	or
`//title/"airplane"`	content
`//title[toc='yes']`	attributes
`//section[_ subsection _]/title`	titles of sections with subtitles
`a[_ b#c _]/d`	context qualifier

Alternatively with grammar patterns.

FXGrep: Pattern Language

```
//section[!_ subsection _][_(//footnote/'automata')_]/title
```

The titles of all sections that do not have a child of type `subsection` and that do have a descendant of type `footnote` which contains the text `automata`.

FXGrep: Theory

- words are monadic trees
- ranked trees are terms with internal nodes being operators and leaves being constants
- unranked trees represent document structure, nodes are elements
- unranked trees can be represented by ranked trees, but not uniquely
- trees can be specified with grammars, regular expressions, formulas in fixpoint or monadic second-order logic, automata
-

specification	automaton
grammar	polynomial
regular expressions	polynomial
fixpoint formulas	exponential
MSO formulas	non-elementary

- Nice animated slides by Helmut Seidl (Uni Trier)

Tree Automata

- ranked case
  - work bottom-up from the frontier to the root
  - transitions according to the states of the daughters and the label of the mother
  - a tree is accepted if it halts in a final state at the root
- unranked case
  - additional word automaton to recognize the sequence of labels of the daughters
- variations
  - extension to “hedges” (Murata, Brüggemann-Klein, Wood)
  - tree-walking automata navigate through the tree
  - caterpillars (Brüggemann-Klein, Herrmann, Wood)
  - pushdown tree automata (Neumann/Seidl)

Query Automata (Neven/Schwentick)

- two-way tree automata + a selection function
  - partition state-alphabet symbol pairs in U and D
  - always works on a cut of the tree
  - down transition: a node is replaced by its children
  - up transition: all children of a node are replaced by the node
  - ➪ new states for all nodes in the cut
- query: the set of nodes selected by the two-way tree automaton via the selection function
- Expressivity: unary MSO patterns
- Extension to the unranked case possible, but complicated

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Intermediate Conclusion

Fundamental Problem

Scylla (Expressivity)

vs.

Charybdis (Undecidability)

Fundamental Problem II

classical	**semistructured**
RA	“Automata”
FO	MSO
SQL	Tree Description language with regular path descriptions

Secondary Edges

Cross-serial dependencies

(. . . weil) der Karl die Maria dem Peter den Hans schwimmen lehren helfen liess

German: Dijck-Sprache—CF

(. . . omdat) Karel Marie Piet Jan liet helpen leren zwemmen

Dutch: $a^n b^n$—CF

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

Cross-serial dependencies

(. . . weil) der Karl die Maria dem Peter den Hans schwimmen lehren helfen liess

German: Dijck-Sprache—CF

(. . . omdat) Karel Marie Piet Jan liet helpen leren zwemmen

Dutch: $a^n b^n$—CF

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

Cross-serial dependencies

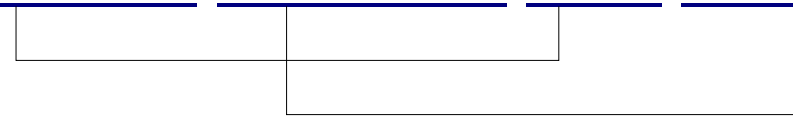
(. . . weil) der Karl die Maria dem Peter den Hans schwimmen lehren helfen liess

German: Dijck-Sprache—CF

(. . . omdat) Karel Marie Piet Jan liet helpen leren zwemmen

Dutch: $a^n b^n$—CF

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



Cross-serial dependencies

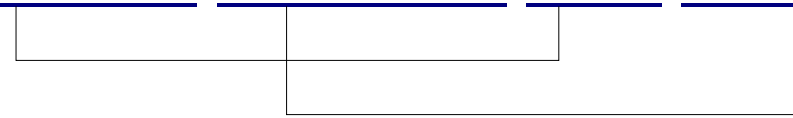
(. . . weil) der Karl die Maria dem Peter den Hans schwimmen lehren helfen liess

German: Dijck-Sprache—CF

(. . . omdat) Karel Marie Piet Jan liet helpen leren zwemmen

Dutch: $a^n b^n$—CF

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



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

Regular Query of context-sensitive relations

Overview: Two-Step Approach

- Assumption: there exists a description of the context-sensitive phenomena (e.g., via a context-free tree grammar)
- database query
  - MSO query into the treebank
    - ➪ set of possible answers/trees
  - lifting of the grammar and the possible answer trees
    - ➪ tree automaton representing the grammar and a set of lifted candidate trees
  - the automaton filters the lifted candidate trees
    - ➪ set of lifted answer trees
- reconstruction of the intended trees via MSO-transduction or macro tree transducer
  - ➪ set of intended answer trees

Context-Free Tree Grammars

A context-free tree grammar Γ is a 5-tuple $\langle \Sigma, \mathbf{F}, S, \mathbf{X}, \mathbf{P} \rangle$, with

$\Sigma, \mathbf{F}$	ranked alphabets of *inoperatives* and *operatives*;
$S \in \mathbf{F}$	start symbol;
$\mathbf{X}$	a set of variables; and
$\mathbf{P}$	a set of productions.

Context-Free Tree Grammars

The productions $p \in \mathbf{P}$ are of the form

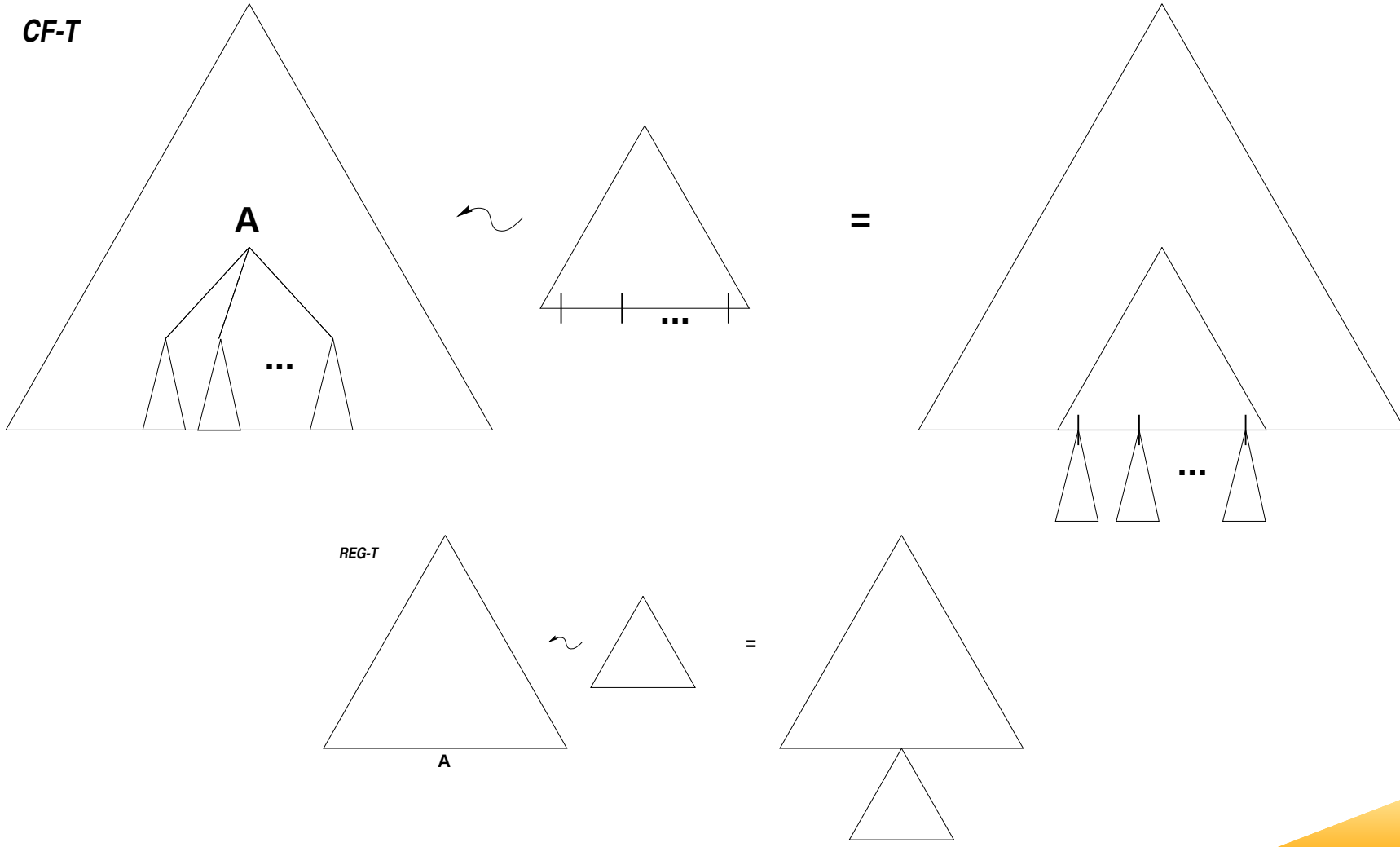
$$F(x_1, \cdots, x_n) \longrightarrow t$$

with $F \in \mathbf{F}$, $x_1, \cdots, x_n \in \mathbf{X}$, and t is a term over $\mathbf{\Sigma} \cup \mathbf{F} \cup \{x_1, \cdots, x_n\}$.

An application of a rule $F(x_1, \cdots, x_n) \longrightarrow t$ "rewrites" a subtree rooted in F as the tree t with its respective variables substituted by F's daughters.

Tree grammars with $\mathbf{F}_n = \emptyset, n \neq 0$, are called *regular*.

Context-Free Tree Grammars



A CFTG for $a^n b^m c^n d^m$: $\mathcal{G}$

$$\Sigma_0 = \{\varepsilon, a, b, c, d\} \qquad \Sigma_2 = \{\bullet\}$$

$$\mathbf{F}_0 = \{S\} \qquad \mathbf{F}_4 = \{F\}$$

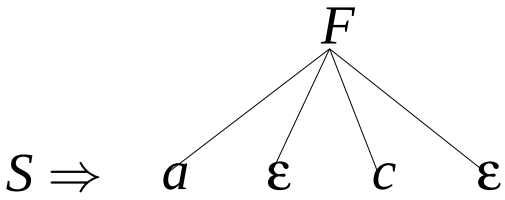
$$S = S \qquad \mathbf{X} = \{x_1, x_2, x_3, x_4\}$$

$$\mathbf{P} = \left\{ \begin{array}{rcl} S & \longrightarrow & \varepsilon \\ S & \longrightarrow & F(a, \varepsilon, c, \varepsilon) \\ S & \longrightarrow & F(\varepsilon, b, \varepsilon, d) \\ F(x_1, x_2, x_3, x_4) & \longrightarrow & F(\bullet(a, x_1), x_2, \bullet(c, x_3), x_4) \\ F(x_1, x_2, x_3, x_4) & \longrightarrow & F(x_1, \bullet(b, x_2), x_3, \bullet(d, x_4)) \\ F(x_1, x_2, x_3, x_4) & \longrightarrow & \bullet(\bullet(\bullet(x_1, x_2), x_3), x_4) \end{array} \right\}$$

An example derivation of *aabccd*

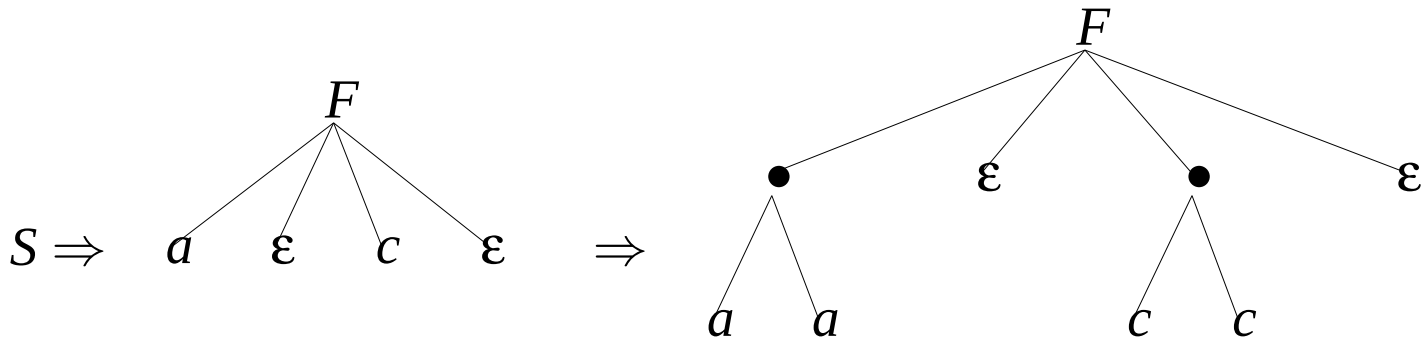
S

An example derivation of *aabccd*



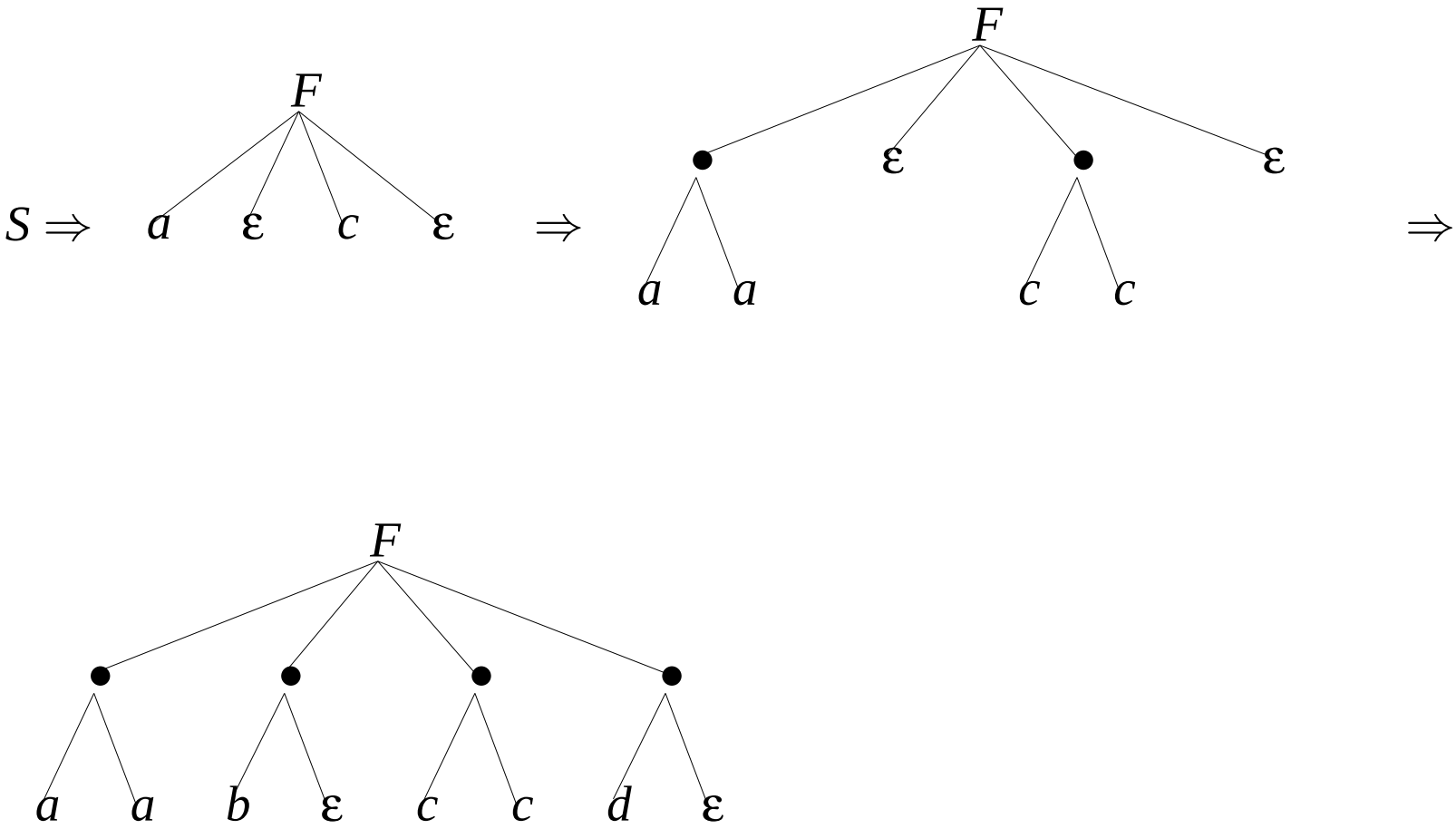
$S \longrightarrow F(a, \varepsilon, c, \varepsilon)$

An example derivation of *aabccd*



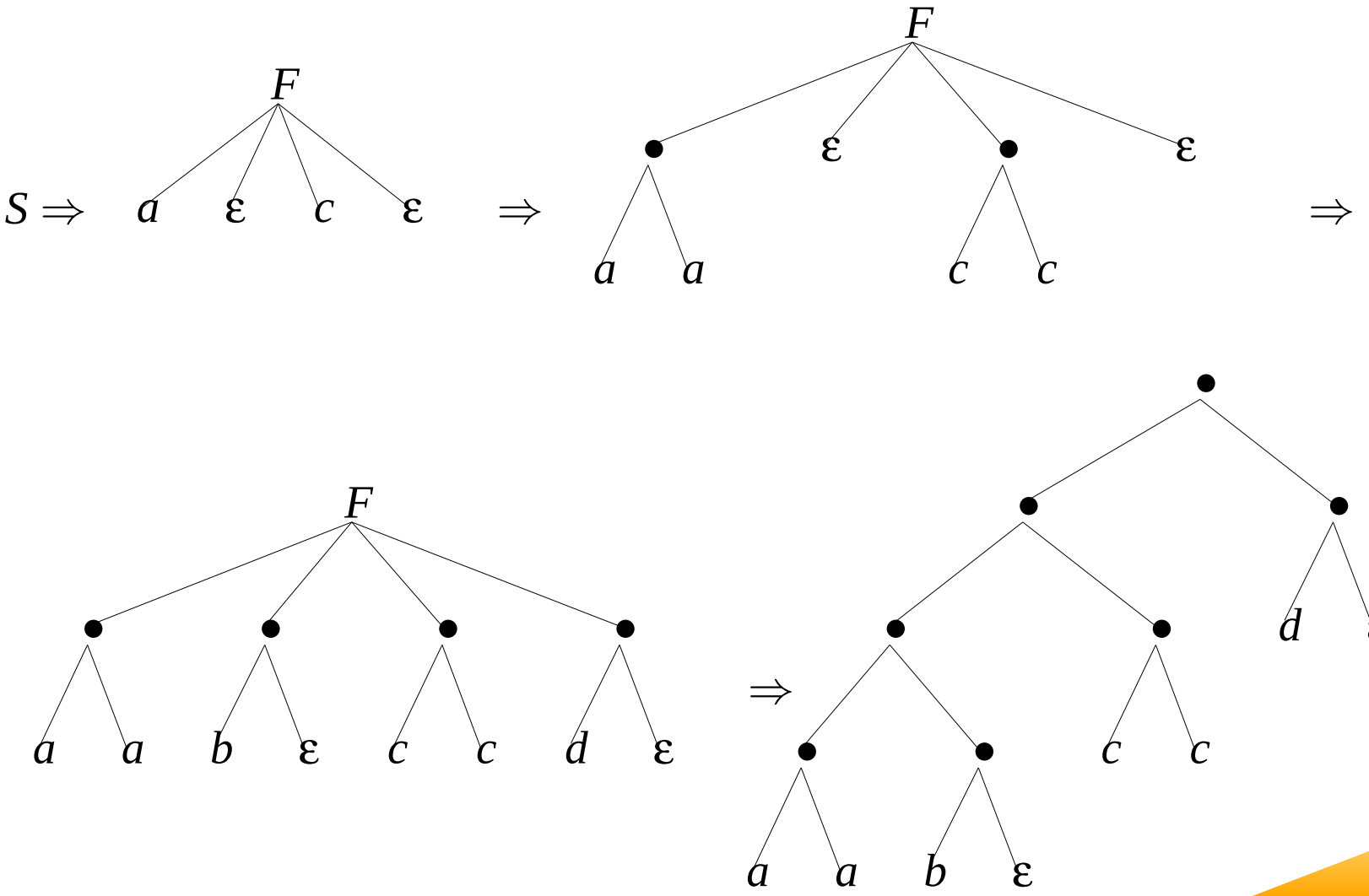
$F(x_1, x_2, x_3, x_4) \longrightarrow F(\bullet(a, x_1), x_2, \bullet(c, x_3), x_4)$

An example derivation of *aabccd*



$F(x_1, x_2, x_3, x_4) \longrightarrow F(x_1, \bullet(b, x_2), x_3, \bullet(d, x_4))$

An example derivation of *aabccd*



$F(x_1, x_2, x_3, x_4) \longrightarrow \bullet(\bullet(\bullet(x_1, x_2), x_3), x_4)$

MSO as a Query Language

MSO Queries

$$Q = \{t \models \varphi_r\}$$

$r \overset{def}{\Longleftrightarrow}$ regular tree expression

$\varphi_r \overset{def}{\Longleftrightarrow}$ translation of r into an MSO formula

Example:

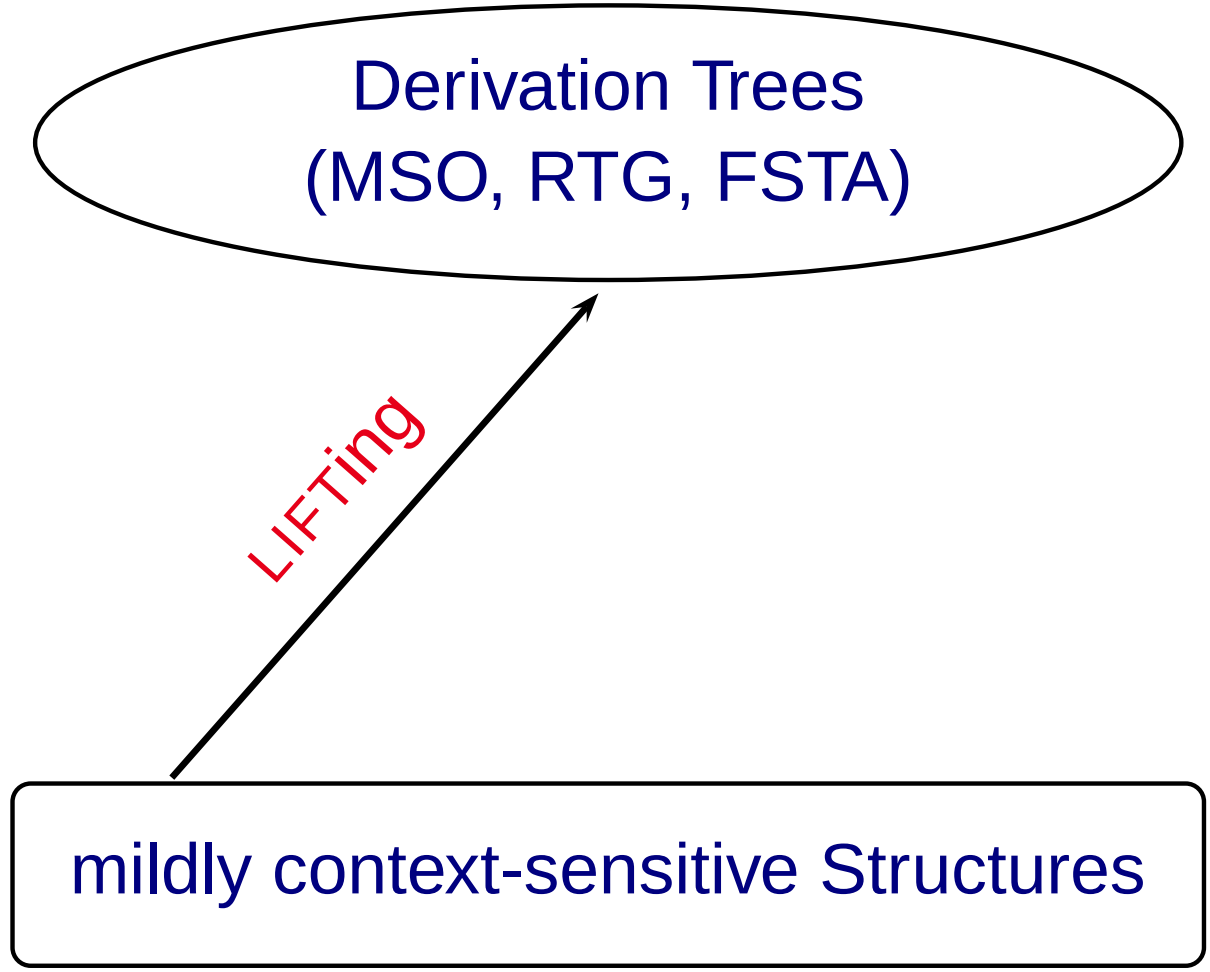
$$(NP_{akk} \cup NP_{dat})^* \cdot \_ \cdot (V_{akk} \cup V_{dat})^*$$

Overview Two-Steps

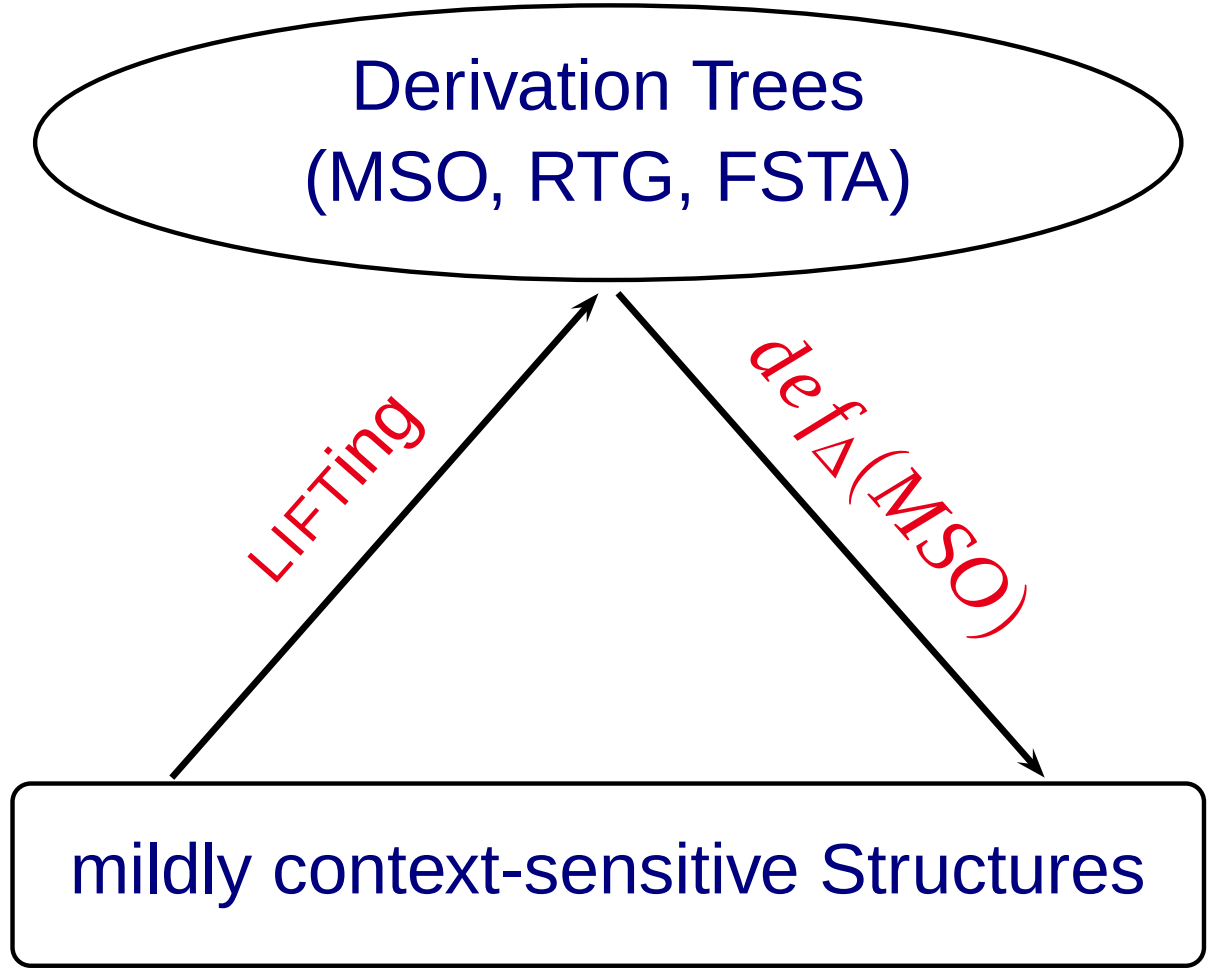
Derivation Trees
(MSO, RTG, FSTA)

mildly context-sensitive Structures

Overview Two-Steps



Overview Two-Steps



Lifting

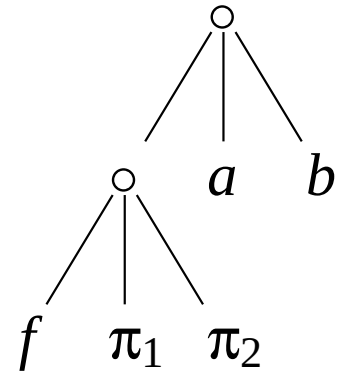
Lifting: Intuition

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

Lifting: Intuition

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

$$f(a,b) \rightsquigarrow (f \circ (\pi_1, \pi_2)) \circ (a,b) \rightsquigarrow$$



Lifting

For $k \geq 0$, $\text{LIFT}_k^{\Sigma} : T(\Sigma, \mathbf{X}_k) \rightarrow T(\Sigma^L, k)$ is defined as follows:

$$\text{LIFT}_k^{\Sigma}(x_i) = \pi_i^k$$

$$\text{LIFT}_k^{\Sigma}(f) = c_{(0,k)}(f') \text{ f"ur } f \in \Sigma_0$$

$$\text{LIFT}_k^{\Sigma}(f(t_1, \ldots, t_n)) = c_{(n,k)}(f', \text{LIFT}_k^{\Sigma}(t_1), \ldots, \text{LIFT}_k^{\Sigma}(t_n)) \text{ f"ur } f \in \Sigma_n, n \geq 1$$

The lifted Regular Tree Grammar: $\mathcal{G}'$

$$\Sigma_0^L = \{\varepsilon, a', b', c', d', S'\} \qquad \Sigma_2^L = \{\bullet'\}$$

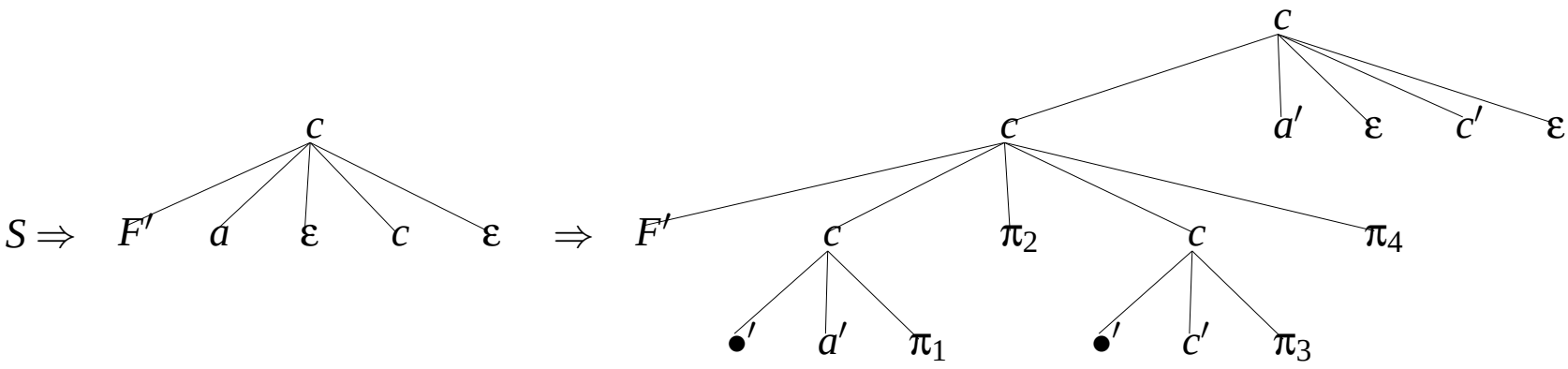
$$\Sigma_4^L = \{F'\} \cup \{\pi_1, \pi_2, \pi_3, \pi_4\} \qquad \Sigma_{n,k}^L = \{c\} \text{ for simplicity}$$

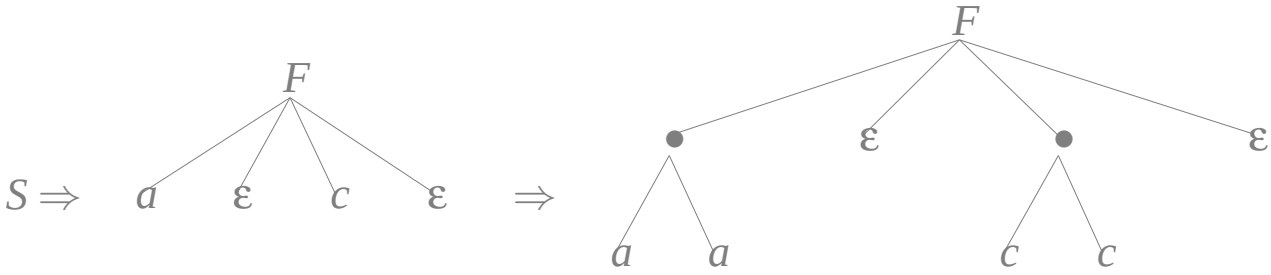
$$\mathbf{F}_0 = \{ S', F' \} \qquad S = S' \qquad \mathbf{X} = \emptyset$$

$$\mathbf{P} = \left\{ \begin{array}{lcl} S' & \longrightarrow & \varepsilon \\ S' & \longrightarrow & c(F', a', \varepsilon, c', \varepsilon) \\ S' & \longrightarrow & c(F', \varepsilon, b', \varepsilon, d') \\ F' & \longrightarrow & c(F', c(\bullet', a', \pi_1), \pi_2, c(\bullet', c', \pi_3), \pi_4) \\ F' & \longrightarrow & c(F', \pi_1, c(\bullet', b', \pi_2), \pi_3, c(\bullet', d', \pi_4)) \\ F' & \longrightarrow & c(\bullet', c(\bullet', c(\bullet', \pi_1, \pi_2), \pi_3), \pi_4) \end{array} \right\}$$

$F(x_1, x_2, x_3, x_4) \longrightarrow F(\bullet(a, x_1), x_2, \bullet(c, x_3), x_4)$

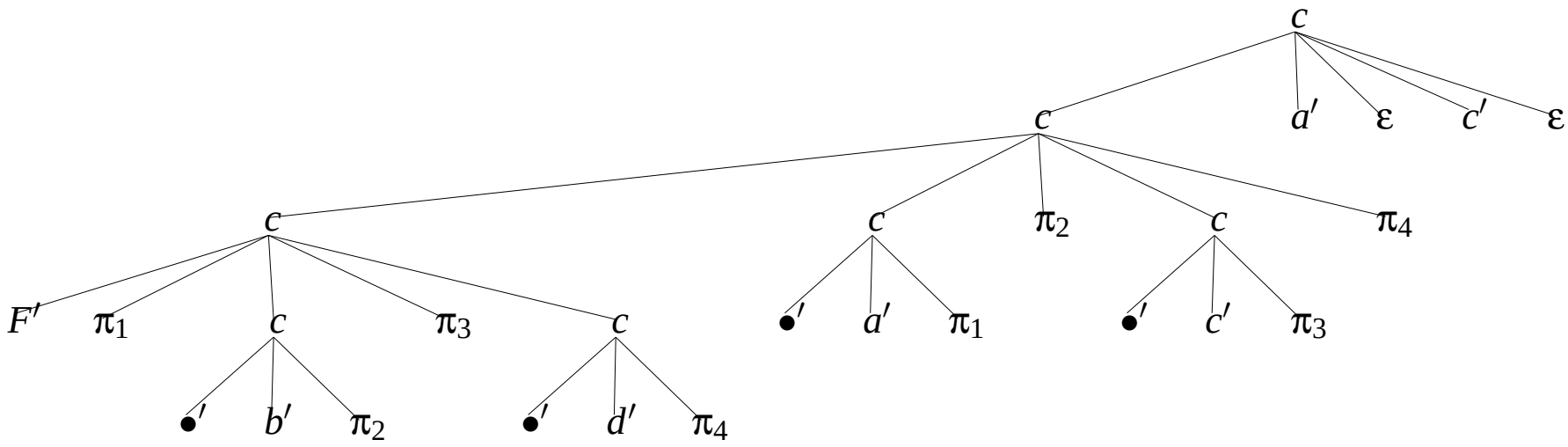
A sample derivation of $\mathcal{G}'$



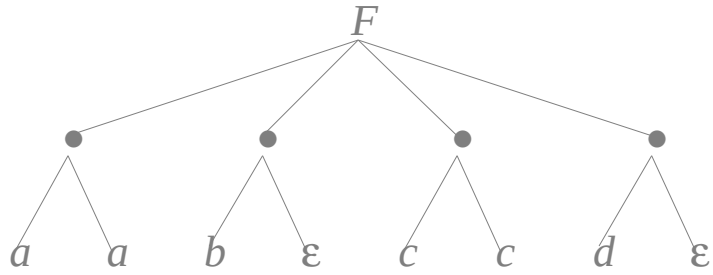


A sample derivation of $\mathcal{G}'$

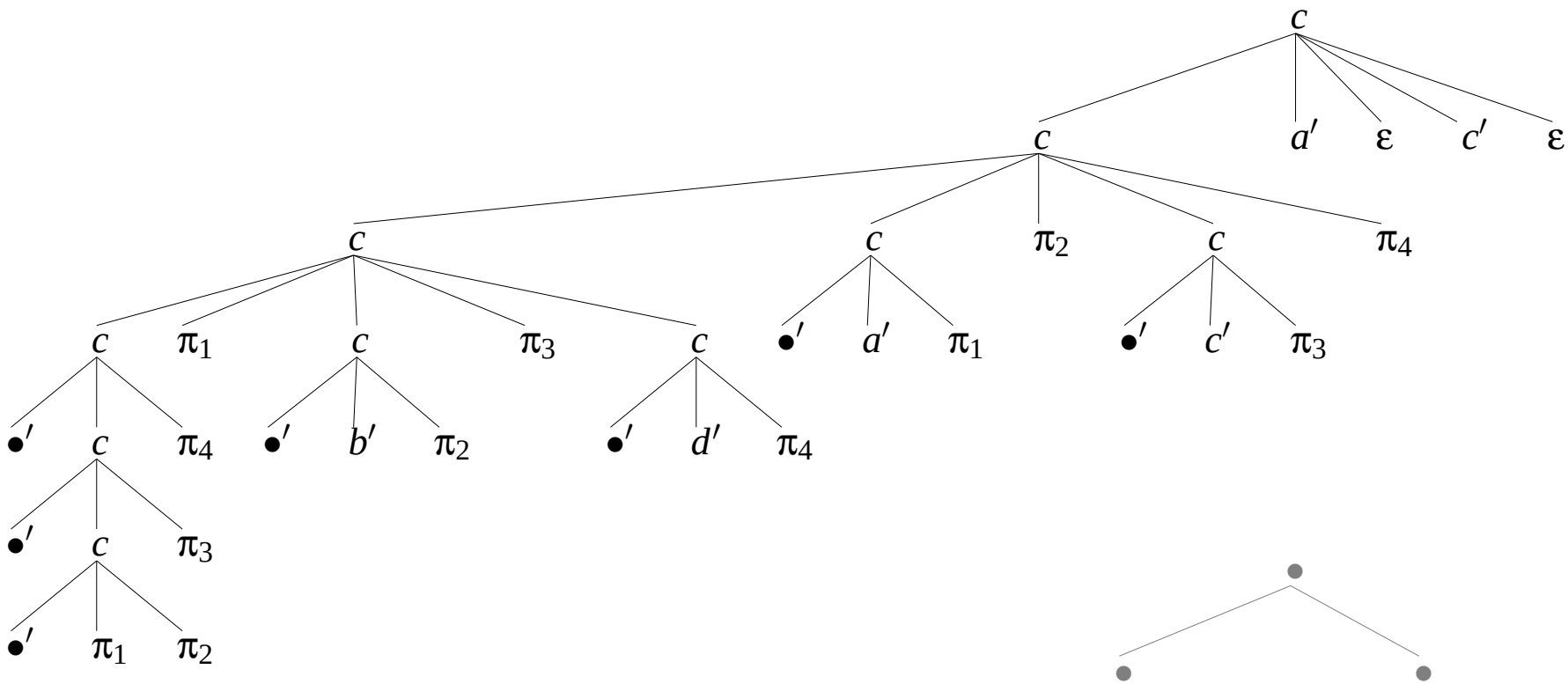
$\Rightarrow$

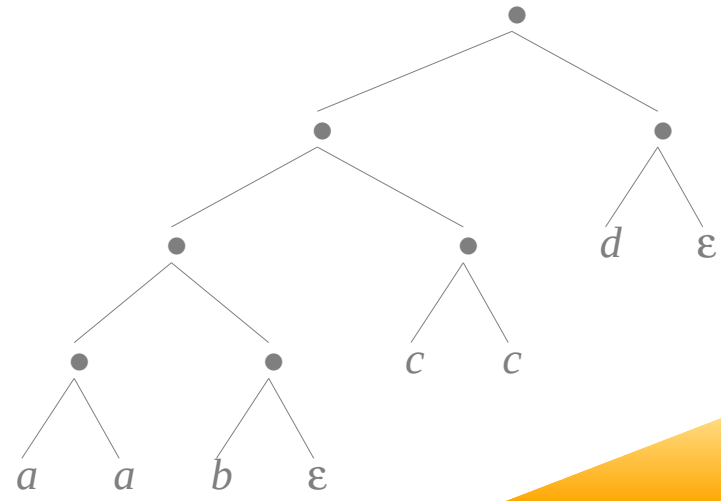


$\Rightarrow$



A sample derivation of $\mathcal{G}'$





Lifted Query

$$Q^L = \{t^L \models \varphi_r^L \wedge \psi\}$$

$\varphi_r^L \overset{def}{\Longleftrightarrow}$ "Lifted" φ_r

$\psi \overset{def}{\Longleftrightarrow}$ "Lifted" specification of cross-serial dependencies

But: the resulting trees are not in the format of the treebank and for humans/linguists unreadable.

Reconstruction

Goal of the Reconstruction

- The intended tree is contained in the lifted tree.
- Homomorphism:

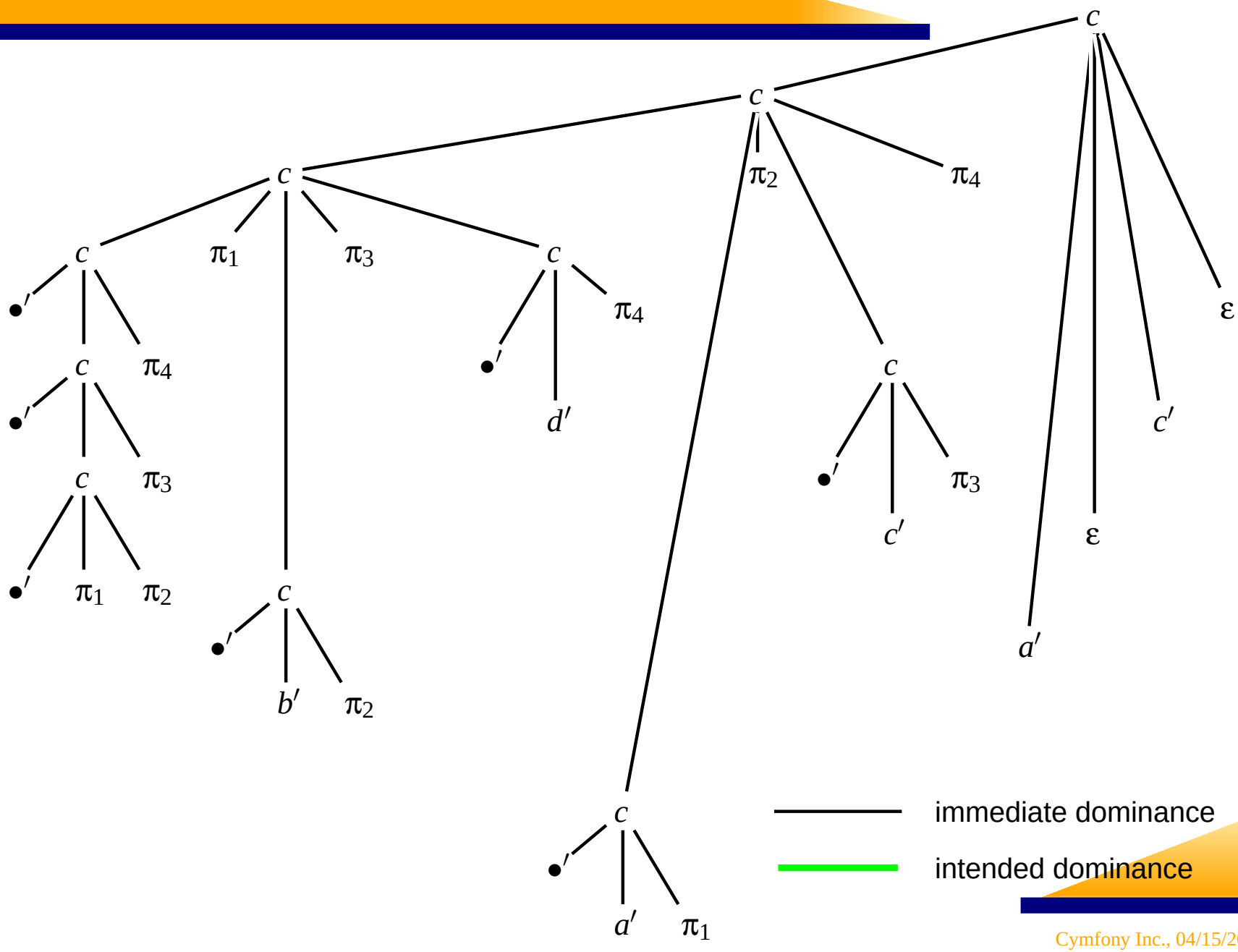
$$h(f') = f(x_1, \ldots, x_n) \text{ for } f \in \Sigma_n$$
$$h(\pi_i^n) = x_i$$
$$h(c_{(n,k)}(t, t_1, \ldots, t_n)) = h(t)[h(t_1), \ldots, h(t_n)]$$

- Definition of a set of intended relations:

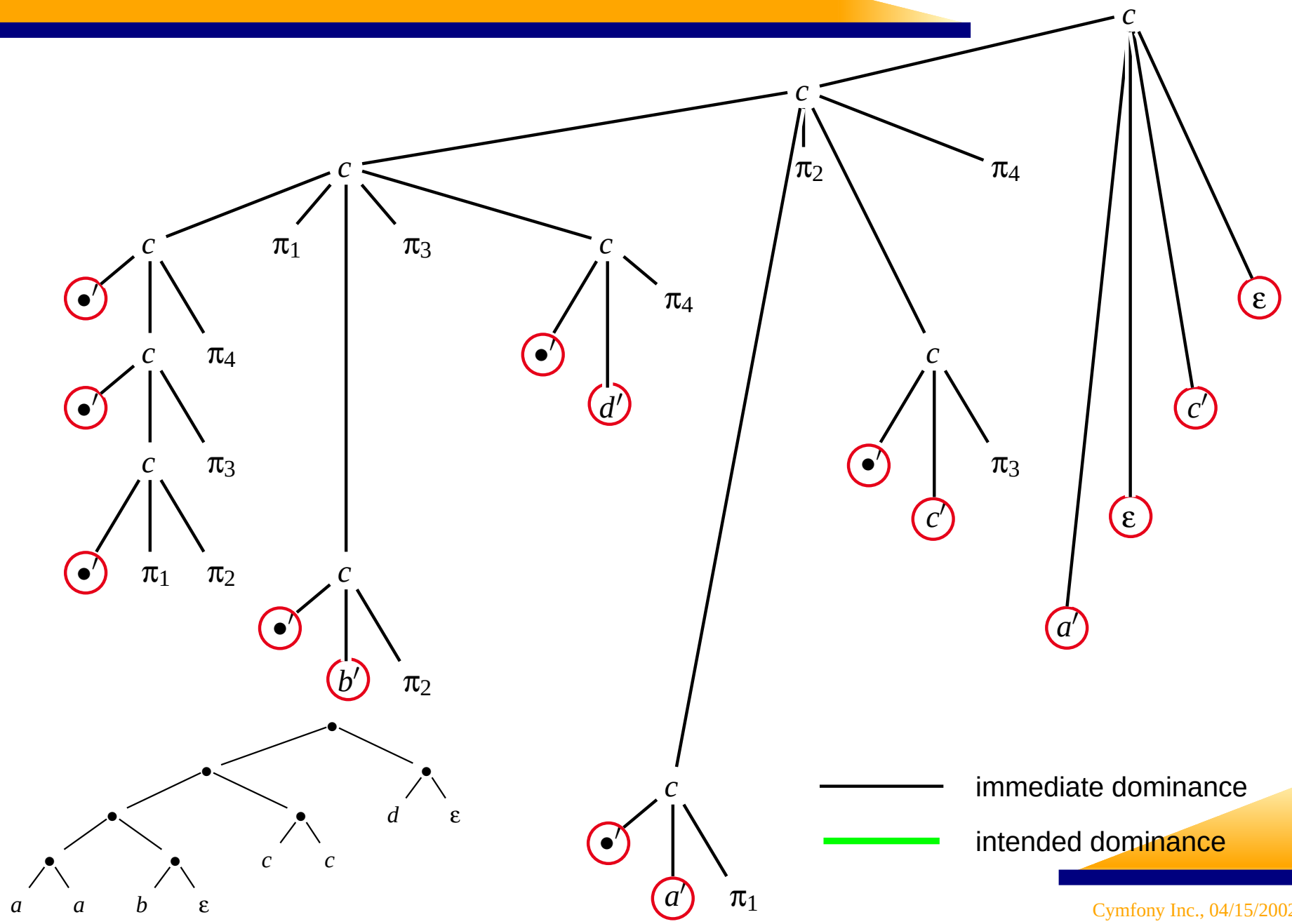
$$R^I = \{\lhd^\star \text{ (dominance)}, \lhd, \lhd^+, \lhd \text{ (precedence)}, \ldots\}$$

➪ MSO-Transduction or Macro-Tree Transducer

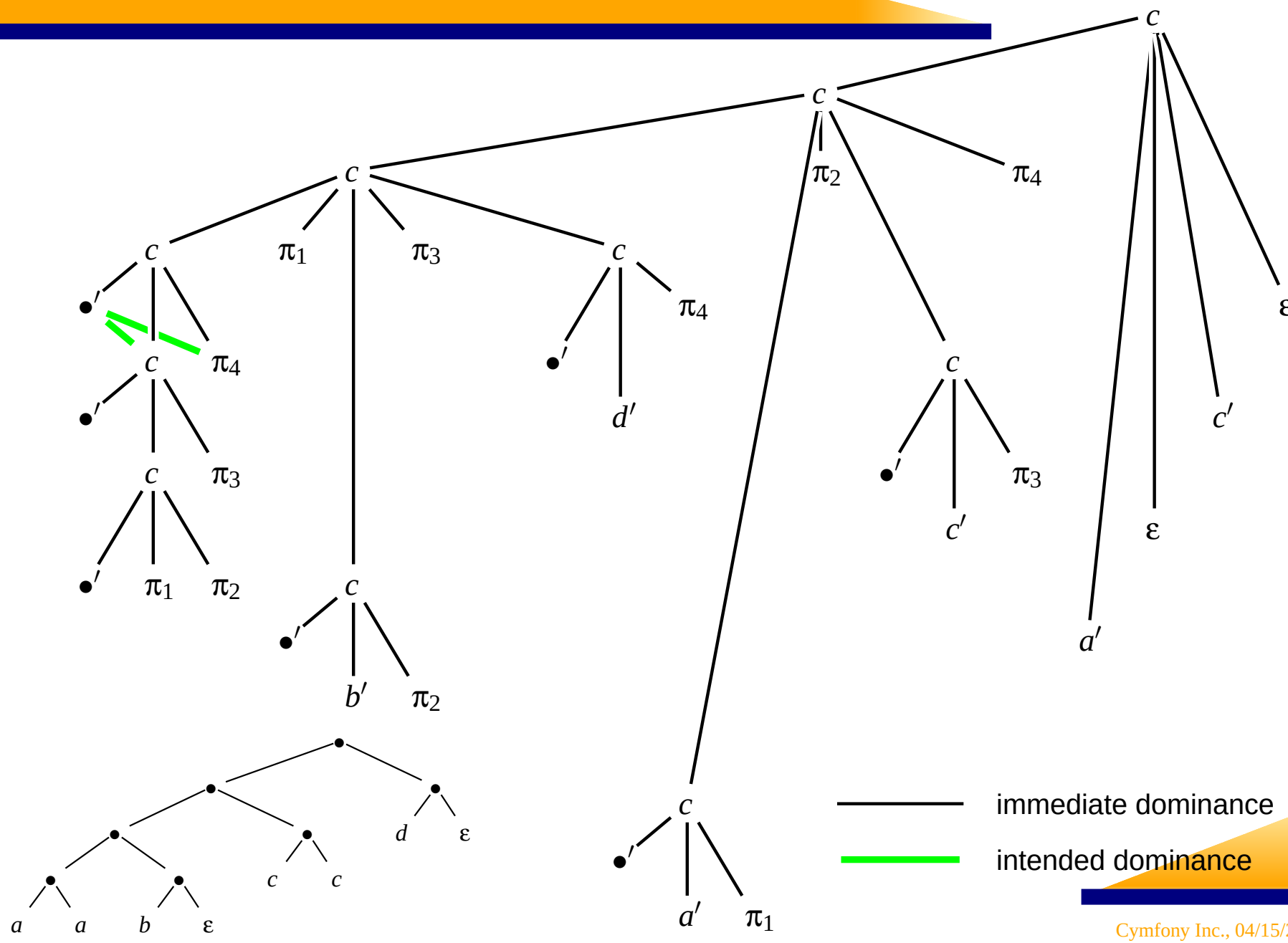
Reconstruction of the Intended Structures



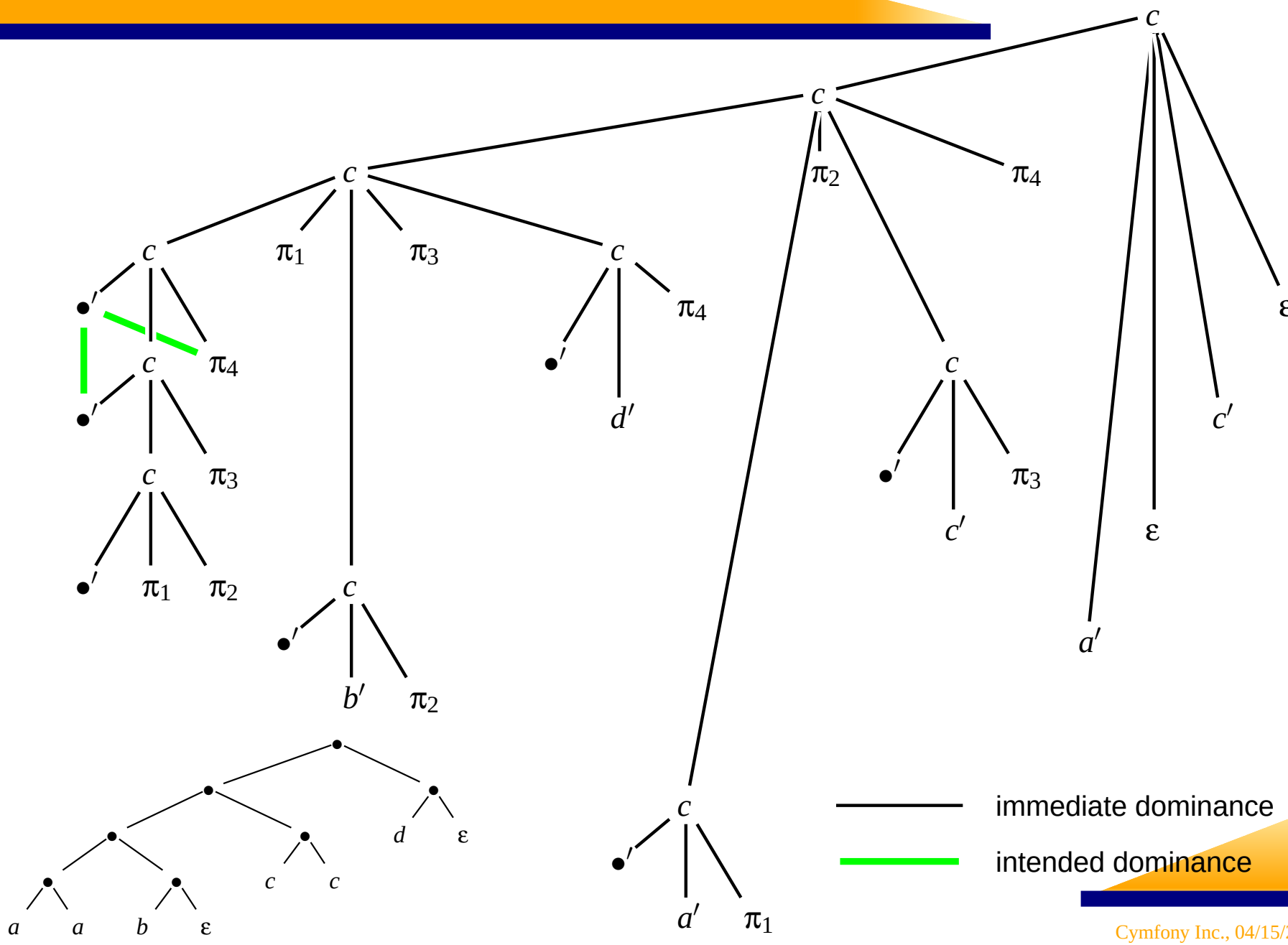
Reconstruction of the Intended Structures



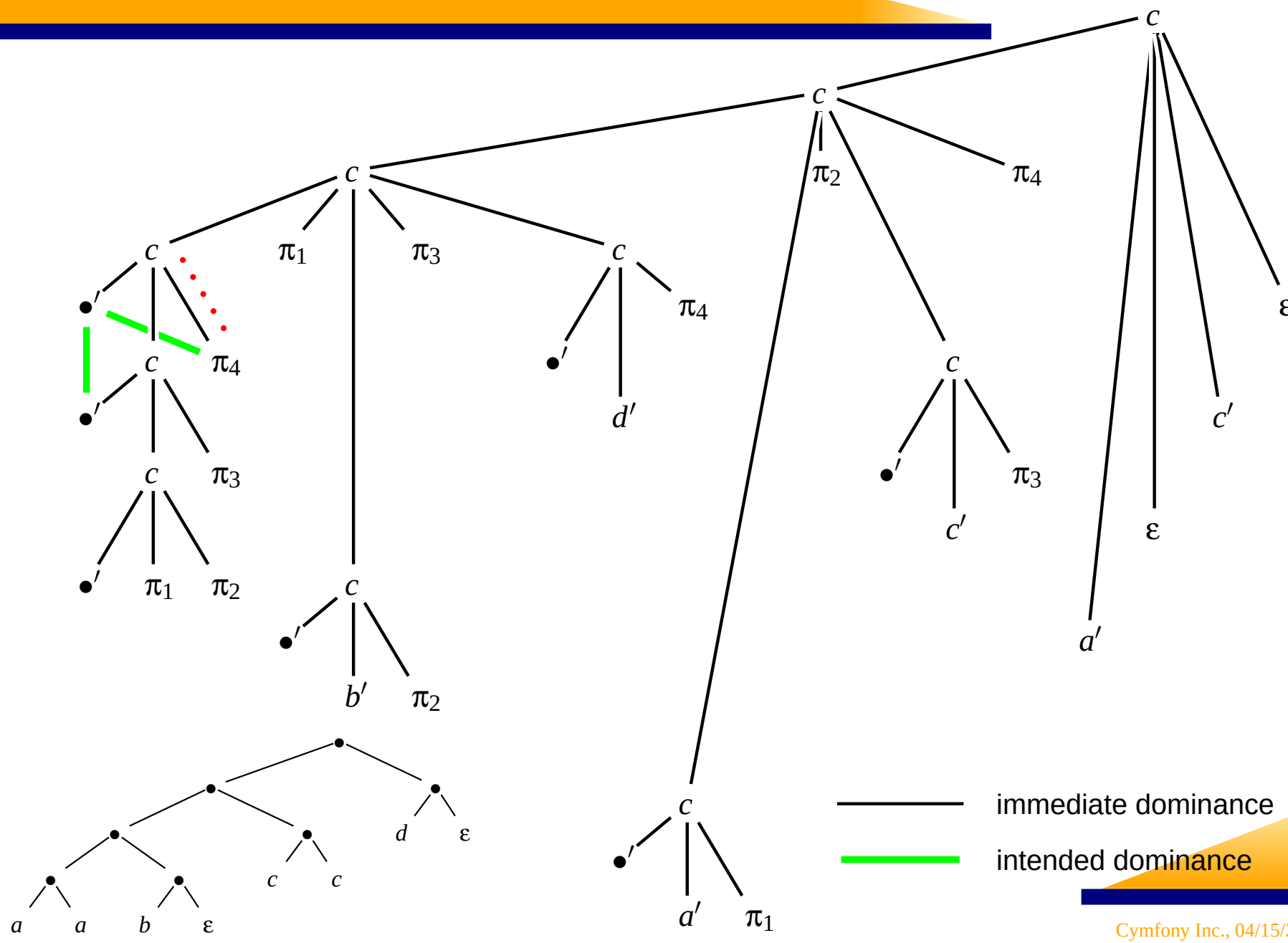
Reconstruction of the Intended Structures



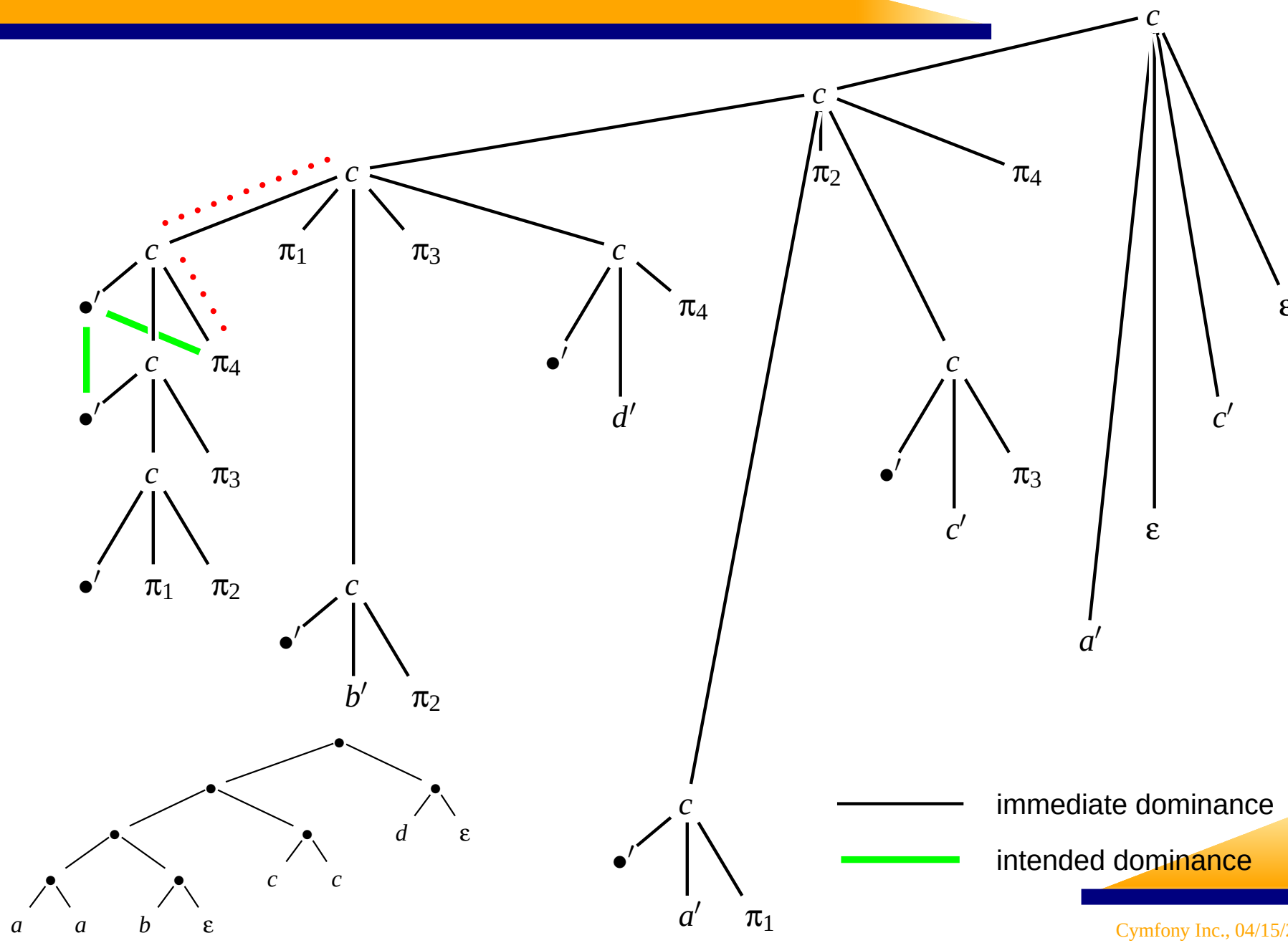
Reconstruction of the Intended Structures



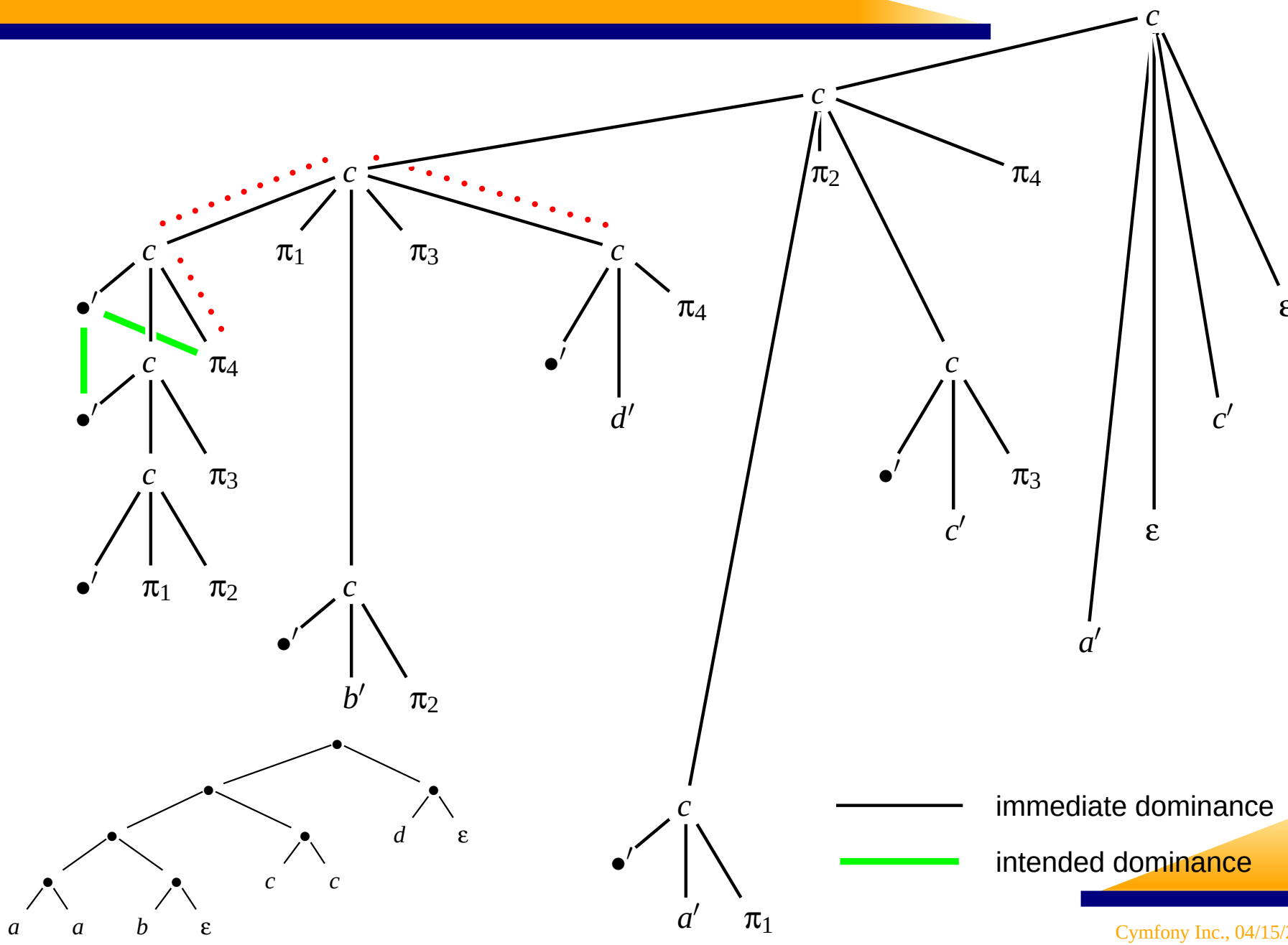
Reconstruction of the Intended Structures



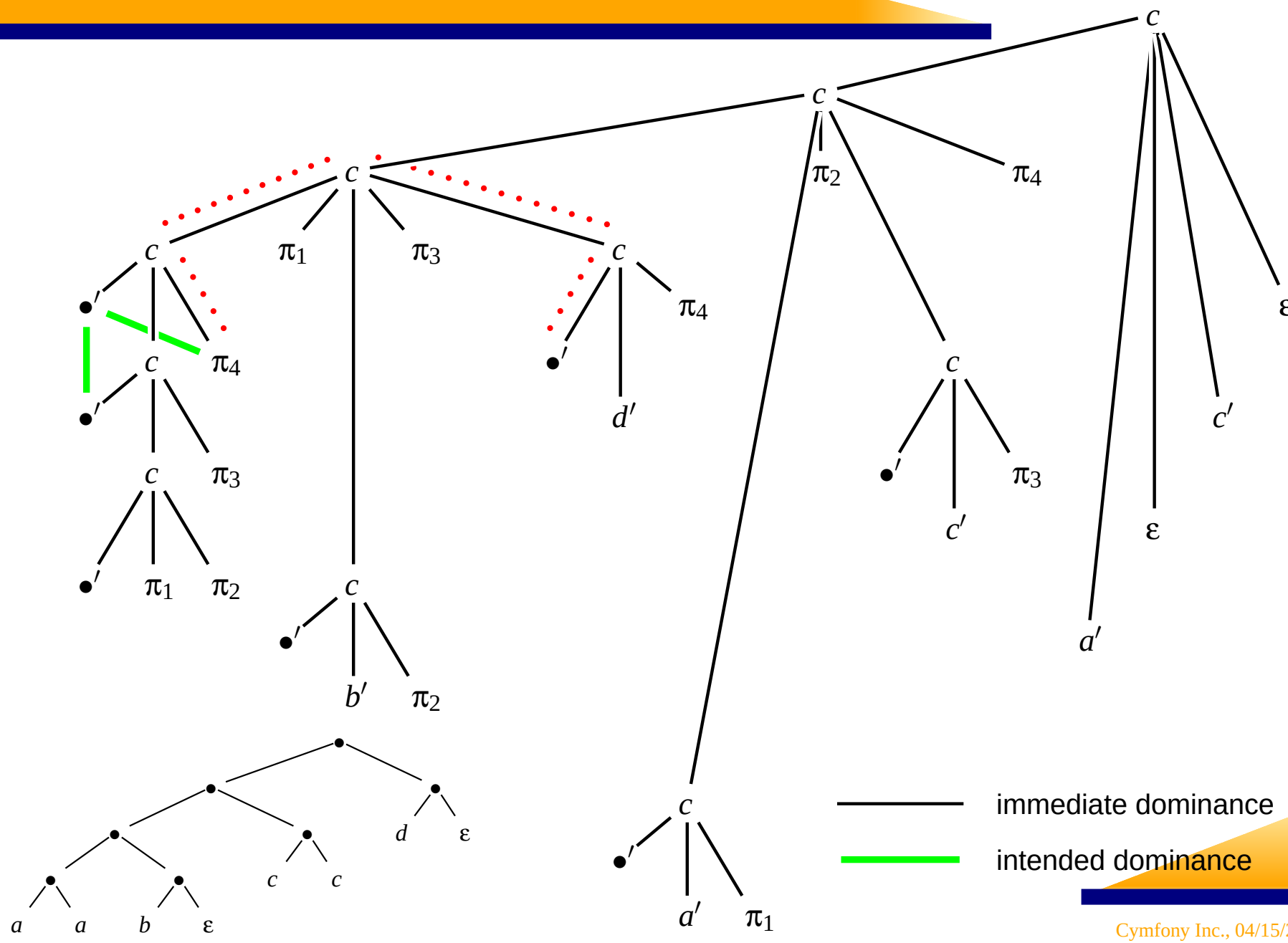
Reconstruction of the Intended Structures



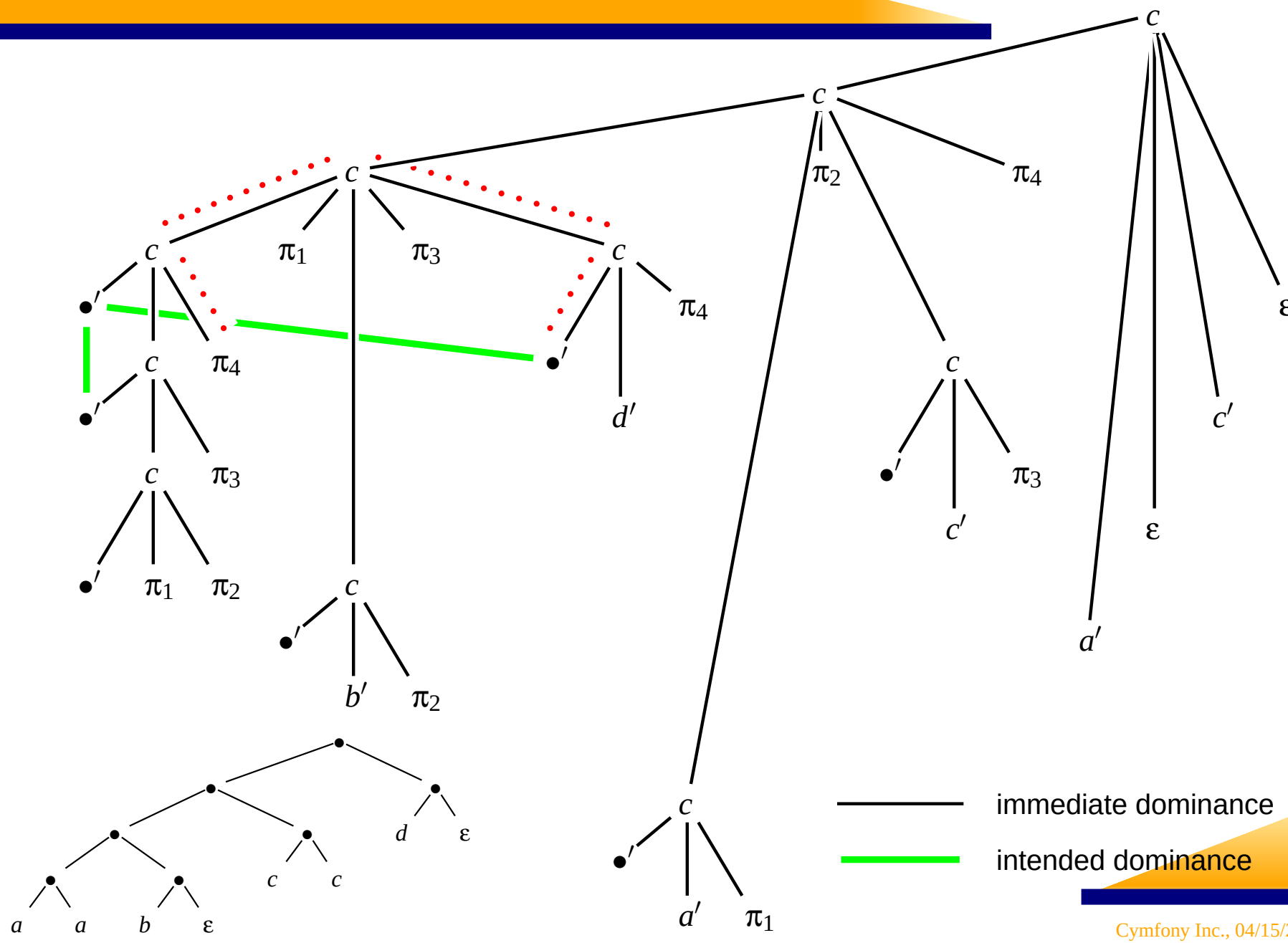
Reconstruction of the Intended Structures



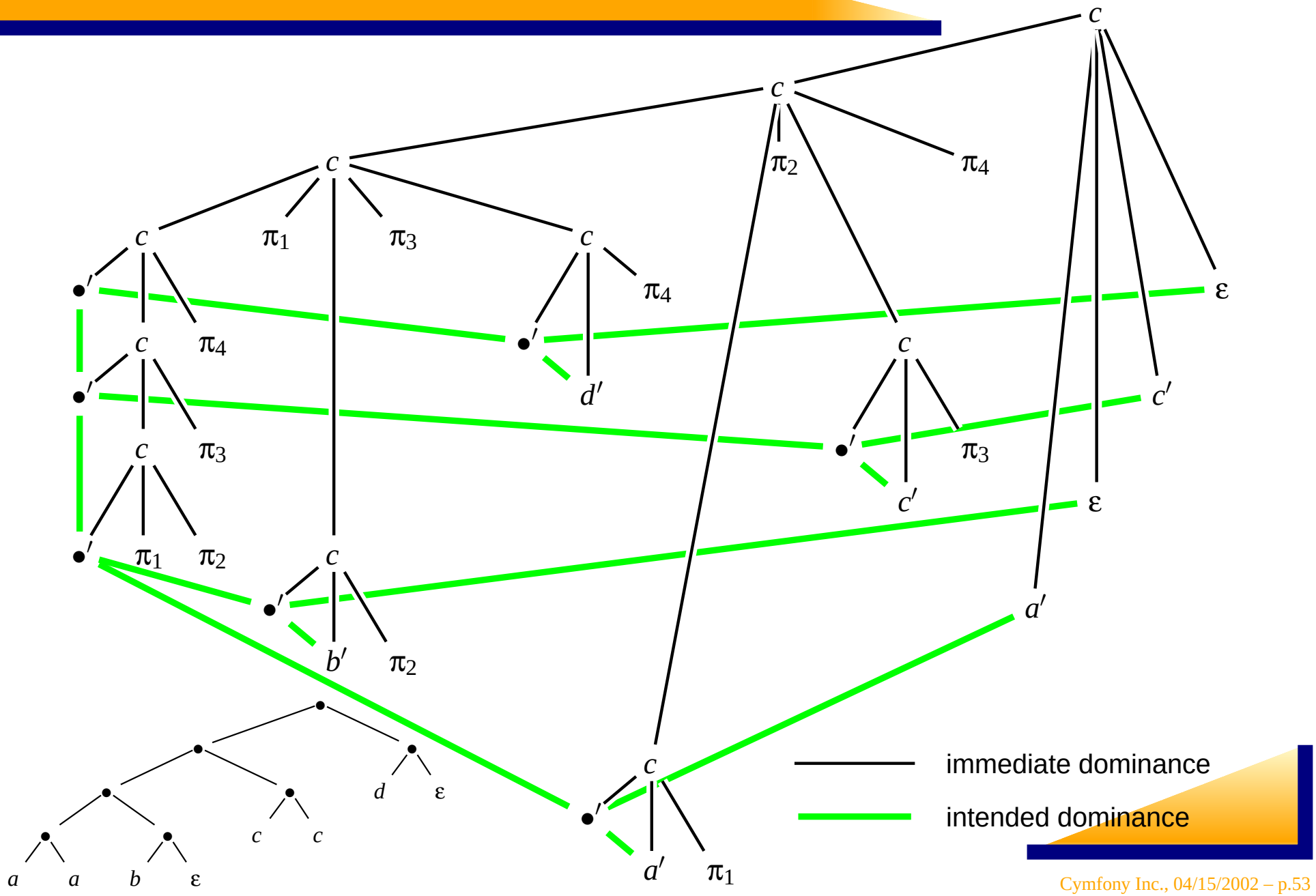
Reconstruction of the Intended Structures



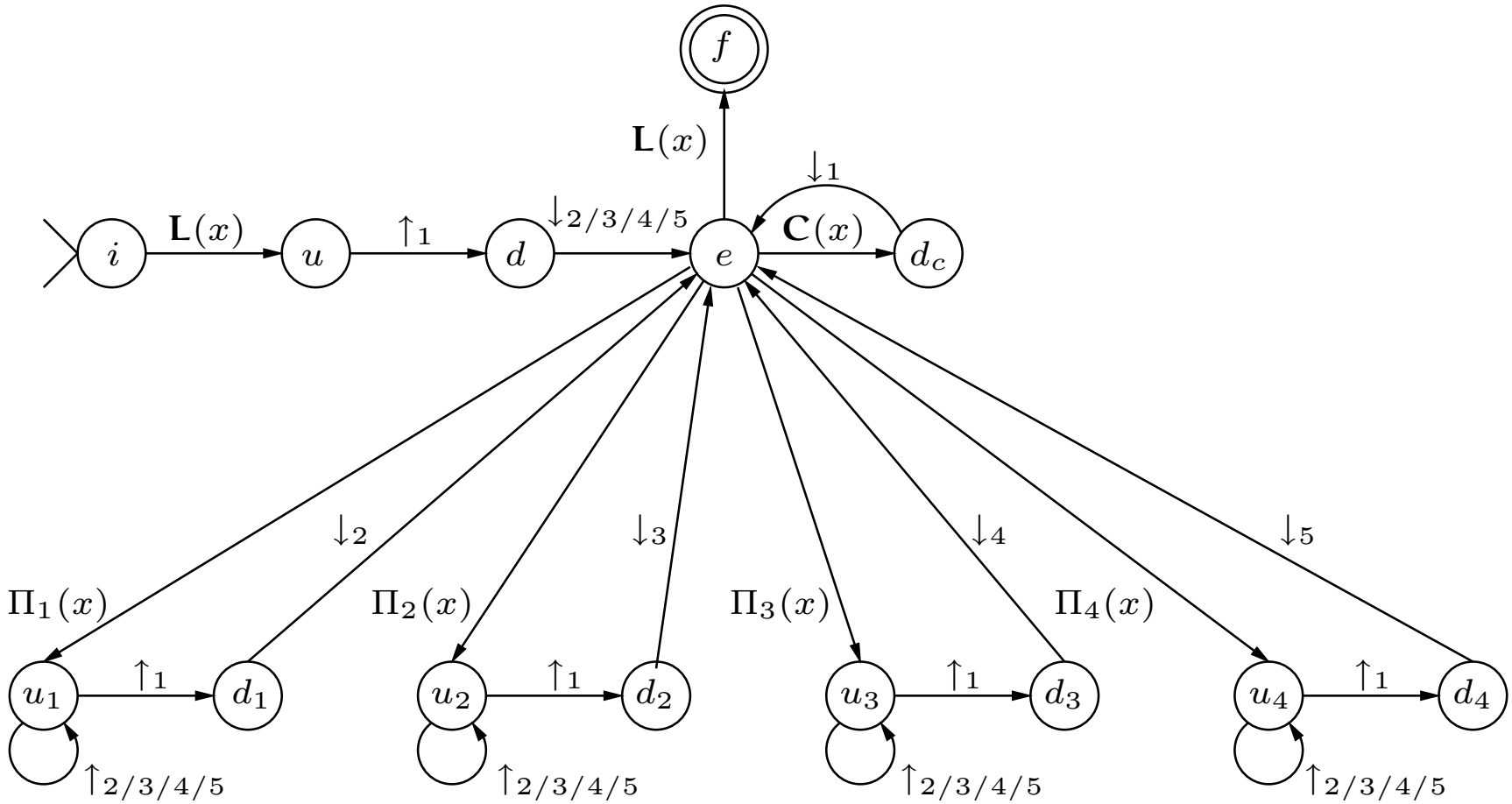
Reconstruction of the Intended Structures



Reconstruction of the Intended Structures



The tree-walking automaton $\mathfrak{A}$



Regular Tree-Node Relations

For any tree t, such a tree-walking automaton $\mathfrak{A}$ computes a node relation

$$R_t(\mathfrak{A}) = \{(x,y) \mid (x,q_i) \stackrel{*}{\Longrightarrow} (y,q_f) \text{ for some } q_i \in I \text{ and some } q_f \in F\}$$

where for all states $q_i, q_j \in Q$ and nodes x, y in t

$$(x,q_i) \Rightarrow (y,q_j) \text{ iff } \exists d \in \Delta : (q_i,d,q_j) \in \delta$$

and y is reachable from x in t via d.

If all the tests $\phi(x)$ of $\mathfrak{A}$ are MSO definable, $\mathfrak{A}$ specifies a *regular tree-node relation*, which is itself MSO definable.

MSO definable Transduction

$$\mathcal{R} \rightsquigarrow \mathcal{S}$$

$$\Delta = (\chi, \xi, (\theta_s)_{s \in \mathcal{S}})$$

χ	the domain of the transduction
ξ	the resulting domain of $\mathcal{S}$
θ_s	the new relations

Definition by Courcelle 1997

Basic Idea Rabin 1965

Intended Structures via MSO transduction

$$(\varphi, \psi, (\theta_s)_{s \in S})$$

$$S = \{\blacktriangleleft, \blacktriangleleft^\star, \blacktriangleleft^+, \triangleleft, \dots\}$$

$$\chi = \varphi_r^L \wedge \psi$$

$$\xi = \text{nodes with a “linguistic” label}$$

$$\theta_{\blacktriangleleft} = \text{trans}_{W_{\blacktriangleleft}}(x,y)$$

$$\theta_{\blacktriangleleft^\star} = (\forall X)[\blacktriangleleft\text{-}\mathit{closed}(X) \wedge x \in X \rightarrow y \in X]$$

$$\theta_{\blacktriangleleft^+} = x \blacktriangleleft^\star y \vee x \not\approx y$$

$$\theta_{\triangleleft} = \text{trans}_{W_{\triangleleft}}(x,y)$$

$$\theta_{\text{label}} = \text{ taken from } R$$

Top-Down Tree Transducer

A top-down tree transducer (TDTT) is a a tuple $T = \langle Q, \Sigma, \Omega, q_0, P \rangle$ with states Q, ranked alphabets Σ and Ω (input and output), initial state q_0 and a finite set of productions P of the form

$$q(\sigma(x_1, \ldots, x_n)) \longrightarrow t$$

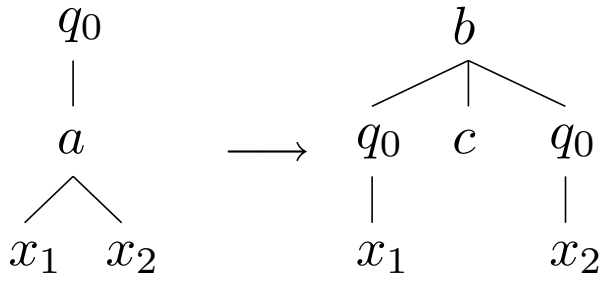
where $n \geq 0$, $\sigma \in \Sigma_n$ and $t \in T(\Omega \cup \Sigma(X) \cup Q)$.

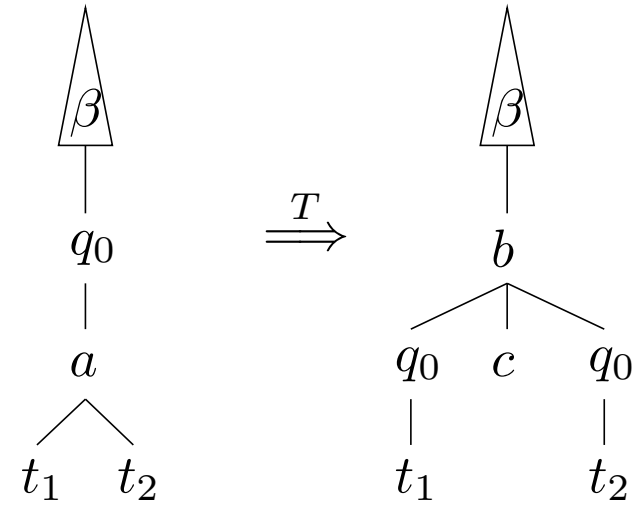
The transition relation ($\overset{T}{\Longrightarrow}$) is defined as usual. The transduction realized by a top-down tree transducer T is then defined to be $\{(t_1, t_2) \in T(\Sigma) \times T(\Omega) \mid q_0(t_1) \overset{T}{\Longrightarrow}^* t_2\}$.

Top-Down Tree Transducer

$$q_0(a(x_1, x_2)) \longrightarrow b(q_0(x_1), c, q_0(x_2))$$

$$q_0(p) \longrightarrow q$$





Macro Tree Transducer

A macro tree transducer (MTT) is a five-tuple $M = \langle Q, \Sigma, \Omega, q_0, P \rangle$ with Q a ranked alphabet of states, ranked alphabets Σ and Ω (input and output), initial state q_0 of rank 1, and a finite set of productions P of the form

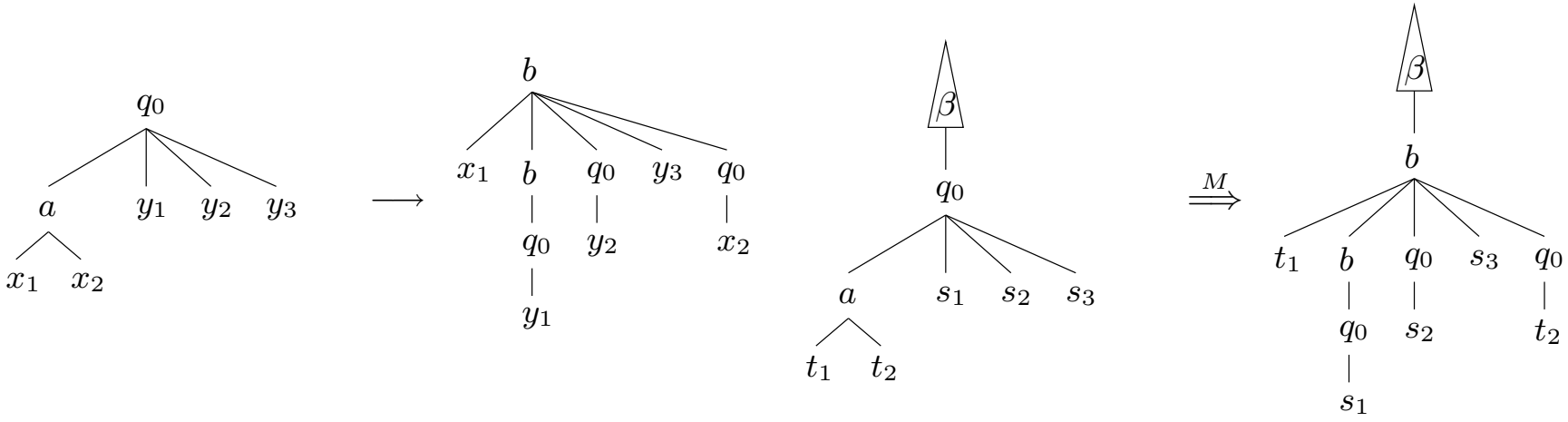
$$q(\sigma(x_1, \ldots, x_n), y_1, \ldots, y_m) \longrightarrow t$$

where $n, m \geq 0$, $q \in Q_{m+1}$, $\sigma \in \Sigma_n$ and $t \in RHS(\Sigma, \Omega, n, m)$.

The productions $p \in P$ of M are used as term rewriting rules in the usual way. The transition relation of M is denoted by $\overset{M}{\Longrightarrow}$. The transduction realized by M is the function $\{(t_1, t_2) \in T(\Sigma) \times T(\Omega) \mid (q_0, t_1) \overset{M}{\Longrightarrow}^* t_2\}$.

Macro Tree Transducer

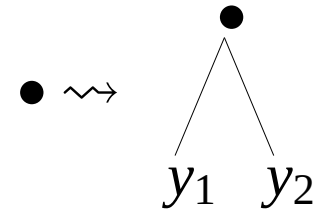
$$q_0(a(x_1,x_2),y_1,y_2,y_3) \longrightarrow b(x_1,b(q_0(y_1)),q_0(y_2),y_3,q_0(x_2))$$

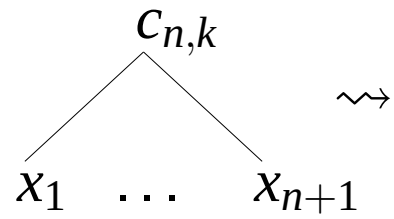


Constructing an MTT: Intuition

$$\sigma \rightsquigarrow \sigma$$

$$\pi_i \rightsquigarrow y_i$$

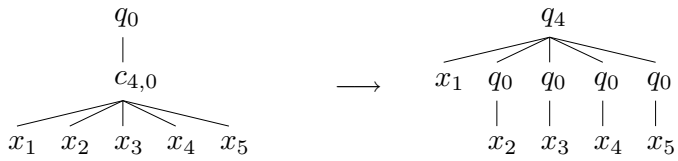


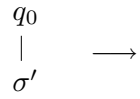


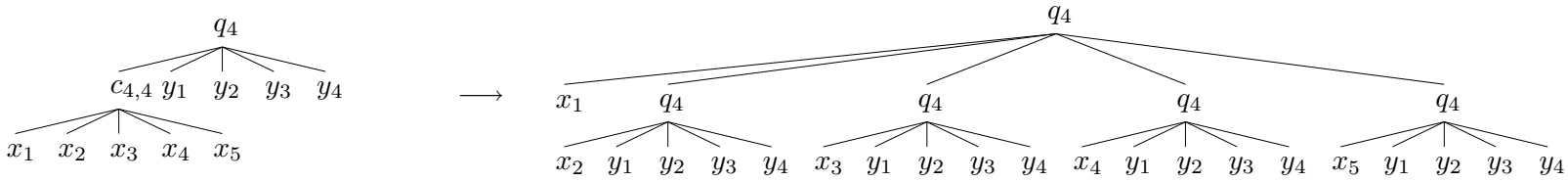
$$q_k(\underbrace{c_{n,k}(x_1,\ldots,x_{n+1}),y_1,\ldots,y_k}_{k+1}) \longrightarrow$$

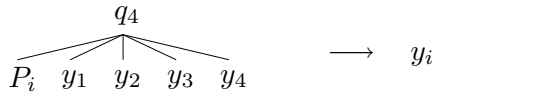
$$q_n(\underbrace{x_1,q_k(x_2,y_1,\ldots,y_k),\ldots,q_k(x_{n+1},y_1,\ldots,y_k)}_{n+1})$$

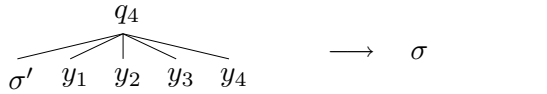
The Example MTT for $\mathcal{G}'$

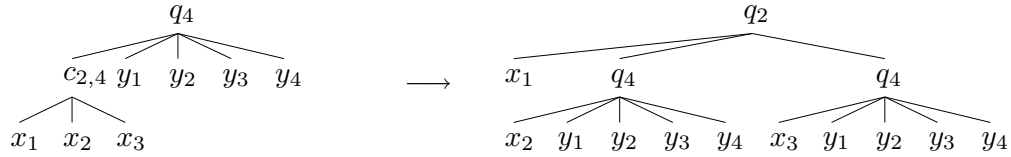












Intended Query-Results

$$Q' = \{def_\Delta(t^L) \mid t^L \models \varphi_r^L \wedge \psi\}$$

$def_\Delta \overset{def}{\Longleftrightarrow}$ transduction from the domain of the lifted regular trees into original tree data base

$\Delta \overset{def}{\Longleftrightarrow}$ MSO definable transduction

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Summary and Outlook

Summary

- Querying context-sensitive relations with regular means.
- three “kinds” of trees
  - trees as elements of a free algebra
  - trees as elements of a free clone (Lawvere-Algebra)
  - trees as relational structures
- transduction is effectively realized via tree-walking automata or macro tree transducer

Outlook

- unranked trees
- implementation
- efficiency
  - MSO decidable on trees
  - MSO undecidable on graphs
  - MSO + one binary relation undecidable on trees
  - MSO → tree automata hyper-exponential
  - “Guarded” MSO → tree automata exponential
  - Lifting and transduction require linear time
  - Lifting of the candidate trees?

Overview

- Introduction and Motivation
- Semistructured Data
- Query Languages
- Intermediate Conclusion
- Secondary Edges
- Regular Queries of Context-Sensitive Relations
- MSO Queries
- Lifting
- Reconstruction
- Conclusion and Outlook

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Appendix
Some Details and Definitions

MSO Logic on Trees (Rogers, 1994)

- K a set of individual constants
- P a set of predicate constants
- immediate dominance: $\lhd$
- reflexive dominance: $\lhd^*$
- proper dominance: $\lhd^+$
- proper precedence: $\prec$
- a set of *monadic* second-order variables ($X, Y, \ldots$);
- a set of individual variables ($x, y, \ldots$);
- boolean connectives: $\neg$, $\wedge$, $\ldots$;
- Quantifiers over individuals and sets: $\exists$, $\forall$;
- equality and membership: $\approx$, $\in$.

Example formulas

$$X \subseteq Y \stackrel{def}{\Longleftrightarrow} (\forall x)[x \in X \Rightarrow x \in Y]$$

$$\text{Sing}(X) \stackrel{def}{\Longleftrightarrow}$$

% All subsets of X are equal to X or empty:

$$(\forall Y)[Y \subseteq X \Rightarrow (X \subseteq Y \vee (\forall Z)[Y \subseteq Z])] \wedge$$

% X is not the empty set:

$$(\exists Y)[X \not\subseteq Y]$$

Example formulas

AC-Com$(x,y) \stackrel{def}{\Longleftrightarrow}$

% x c-commands y:

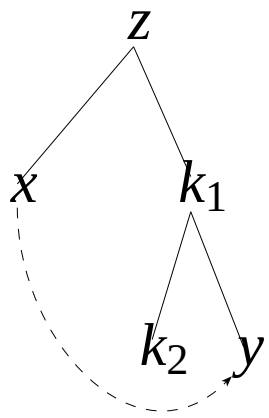
$(\forall z)[z \lhd^{+} x \Rightarrow z \lhd^{+} y] \wedge \neg(x \lhd^{*} y) \wedge$

% y does not c-command x:

$\neg((\forall z)[z \lhd^{+} y \Rightarrow z \lhd^{+} x] \wedge \neg(y \lhd^{*} x)) \wedge$

% x precedes y:

$x \prec y$



Example formulas

$$\mathrm{LC}(x,y) \stackrel{def}{\Longleftrightarrow}$$

$$\mathrm{AC\text{-}Com}(x,y) \wedge$$

% There is no k_1 with property P:

$$(\neg\exists k_1)[k_1 \in P \wedge$$

% such that it intervenes between x and y:

$$(\exists z)[z \lhd x \wedge z \lhd^{+} k_1 \wedge k_1 \lhd^{+} y]]$$

Properties of MSO Logic on Trees

- Advantages:
  - Concise Representation in a Powerful and Flexible Logic
  - Descriptive Complexity Results
  - Operational and Denotational Semantics
  - Decidable
  - Compilation: Formulas to Tree Automata
- Shortcomings
  - Non-Elementary Complexity of the Decidability Algorithm:

$$|A| = 2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}} \Big\} m$$

  - Descriptive Complexity: Only Context-Free String Languages

“Linguistic” Definitions

- A constraint is **satisfiable** iff it results in a non-empty automaton.
- A constraint is **valid** iff it results in the trivial non-empty automaton.
- A (maximally) binary branching **tree** is a subset T of the domain of MSO Logic on Trees. A **labeled tree** is a tuple $\langle T, F_1, \ldots, F_k \rangle$.
- A **grammar** in this setting is a definition of a $k+1$-ary relation in MSO Logic on Trees designating all and only the well-formed labeled trees.
- The **recognition problem** is the determinization of the emptiness of the conjunction of the grammar formula/automaton with the input formula/automaton.
- The **parsing problem** is the conjunction of the grammar formula/automaton with the input formula/automaton.

Coding RTGs in MSO Logic (Thomas 1997)

- Define a tree automaton $\mathfrak{A}_{\mathcal{G}'}$ from the RTG $\mathcal{G}'$
- Code its behaviour in $\varphi_{\mathfrak{A}_{\mathcal{G}'}}$

$$(\exists X_0, \ldots, X_m)[\bigwedge_{i \neq j} (\neg\exists y)[y \in X_i \wedge y \in X_j] \wedge$$

$$(\forall x)[(\neg\exists y)[x \lhd y] \rightarrow x \in X_0] \wedge \qquad \text{\% Initial State}$$

$$\bigwedge_{1 \leq n \leq m} (\forall x_1, \ldots, x_n, y)[\bigvee_{\substack{(i_1,\ldots,i_n,\sigma,j) \in \alpha \\ 1 \leq k \leq n}} x_k \in X_{i_k} \wedge y \lhd x_k \wedge y \in X_j \wedge y \in P_\sigma]$$

$$\bigvee_{i \in F} (\exists x \forall y)[x \lhd^* y \wedge x \in X_i] \qquad \text{\% Root}$$

Lifting

Suppose that Σ is a ranked alphabet. The *derived* $\mathbb{N}$-sorted alphabet Σ^L is defined as follows:

For each $n \geq 0$,

$$\Sigma'_n = \{ f' \mid f \in \Sigma_n \}$$

is a new set of symbols of sort n;

back to Lifting

Lifting

Suppose that Σ is a ranked alphabet. The *derived* $\mathbb{N}$-sorted alphabet Σ^L is defined as follows:

for each $n \geq 1$ and each $i, 1 \leq i \leq n$,

$$\pi_i^n$$

is a new symbol, the ith projection symbol of sort n;

back to Lifting

Lifting

Suppose that Σ is a ranked alphabet. The *derived* $\mathbb{N}$-sorted alphabet Σ^L is defined as follows:

for each $n \geq 0, k \geq 0$ the new symbol

$$c_{(n,k)}$$

is the (n,k)th composition symbol.

back to Lifting

Lifting

Suppose that Σ is a ranked alphabet. The *derived* $\mathbb{N}$-sorted alphabet Σ^L is defined as follows:

$$\Sigma^L_0 = \Sigma'_0$$
$$\Sigma^L_n = \Sigma'_n \cup \{\pi^n_i | 1 \leq i \leq n\} \text{ for } n \geq 1$$
$$\Sigma^L_{n,k} = \{c_{(n,k)}\} \text{ for } n, k \geq 0$$
$$\Sigma^L_i = \emptyset \text{ otherwise}$$

back to Lifting

Model-Theoretic Interpretation

Basic Idea (Rabin 1965)

Obtaining a structure $\mathbb{B} = \langle B, Q \rangle$ from a structure $\mathbb{A} = \langle A, \mathcal{R} \rangle$ where $\mathcal{R}$ and Q are families of relation symbols.

back to MSO transduction

Tree-Walking Automaton

A *tree-walking automaton (with tests)* (Bloem and Engelfriet 1997) over some ranked alphabet Σ is a finite automaton $\mathfrak{A} = (Q, \Delta, \delta, I, F)$ with states Q, directives Δ, transitions $\delta : Q \times \Delta Q$ and the initial and final states $I \subseteq Q$ and $F \subseteq Q$ which traverses a tree using three kinds of directives:

- $\uparrow_i$ "move up to the mother of the current node (if it has one and it is its i-th daughter)",
- $\downarrow_i$ "move to the i-th daughter of the current node (if it exists)",
- $\phi(x)$ "verify that ϕ holds at the current node".

back to the dominance automaton

Reflexive Closure in *MSO*

$$\textit{R-closed}(X) \stackrel{def}{\Longleftrightarrow} (\forall x,y)[x \in X \wedge R(x,y) \rightarrow y \in X]$$

$$R^*(x,y) \stackrel{def}{\Longleftrightarrow} (\forall X)[\textit{R-closed}(X) \wedge x \in X \rightarrow y \in X]$$

back to the MSO transduction

Walking language $\Rightarrow$ *MSO*-formula

$$\begin{aligned}
trans_{\emptyset}(x,y) &= \bot \\
trans_{\downarrow_i}(x,y) &= edge_i(x,y) \\
trans_{\uparrow_i}(x,y) &= edge_i(y,x) \\
trans_{\phi(x)}(x,y) &= \phi(x) \wedge x = y \\
trans_{W_1 \cup W_2}(x,y) &= trans_{W_1}(x,y) \vee trans_{W_2}(x,y) \\
trans_{W_1 \cdot W_2}(x,y) &= \exists z \left(trans_{W_1}(x,z) \wedge trans_{W_2}(z,y) \right) \\
trans_{W^*}(x,y) &= trans^*_W(x,y) \\
trans^*_W(x,y) &= \forall X \left(\forall v,w \left(v \in X \wedge trans_W(v,w) \rightarrow w \in X \right) \right. \\
&\qquad \left. \wedge\, x \in X \rightarrow y \in X \right)
\end{aligned}$$

back to the MSO transduction

Complex Right-Hand-Sides

Let Σ and Ω be ranked alphabets and $n, m \geq 0$. The set of right hand sides $RHS(\Sigma, \Omega, n, m)$ over Σ and Ω with n variables and m parameters is the smallest set $rhs \subseteq T(\Sigma \cup \Omega, X_n \cup Y_m)$ such that

1. $Y_m \subseteq rhs$
2. For $\omega \in \Omega_k$ with $k \geq 0$ and $\varphi_1, \ldots, \varphi_k \in rhs$, $\omega(\varphi_1, \ldots, \varphi_k) \in rhs$
3. For $q \in Q_{k+1}$ with $k \geq 0$, $x_i \in X_n$ and $\varphi_1, \ldots, \varphi_k \in rhs$, $q(x_i, \varphi_1, \ldots, \varphi_k) \in rhs$

back to MTTs