# How to split Recursive Automata

Isabelle Tellier

Inria-Lille et LIFO
Université d'Orléans

# Grammatical Inference from Positive Examples

## Available Information

- a set of positive examples
- the target class

## First possible strategy : learning by generalization

- build a least general grammar generating the examples
- apply a generalization operator until it belongs to the target class

## Second possible strategy : learning by specialization

- the initial hypothesis space is the whole target class
- use the examples to constrain this space until it is reduced to one grammar

Overview of known results

| class of languages is a subclass of | regular languages | CF languages |
|---|---|---|
| representation | finite state automata | Categorial Grammars |

Overview of known results

| class of language is a subclass of | regular languages | CF languages |
|---|---|---|
| representation | finite state automata | Categorial Grammars |
| generalization strategy | state fusion (Angluin 81) | unification of categories (Kanazawa 96, 98) |

The links between them : in (Tellier 05, 06)

Overview of known results

| | | |
|---|---|---|
| class of languages is a subclass of | regular languages | CF languages |
| representation | finite state automata | Categorial Grammars |
| generalization strategy | state fusion (Angluin 81) | unification of categories (Kanazawa 96, 98) |
| specialization strategy | state fission (Fredouille 00) | constraints introduction (Moreau 04) |

Overview of known results

| class of languages is a subclass of | regular languages | CF languages |
|---|---|---|
| representation | finite state automata | Categorial Grammars |
| generalization strategy | state fusion (Angluin 81) | unification of categories (Kanazawa 96, 98) |
| specialization strategy | state fission (Fredouille 00) | constraints introduction (Moreau 04) |

The links between them : this paper !

# Outline

1. Introduction
2. Categorial Grammars and Recursive Automata
3. Learning by specialization in both representations
4. Learning from Typed Examples : a new interpretation
5. Conclusion

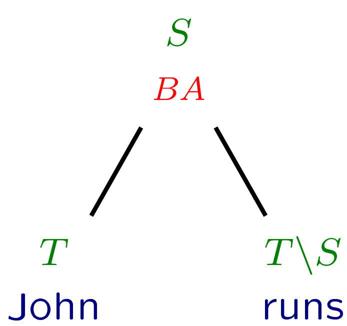
## Definition of a AB-Categorial Grammar

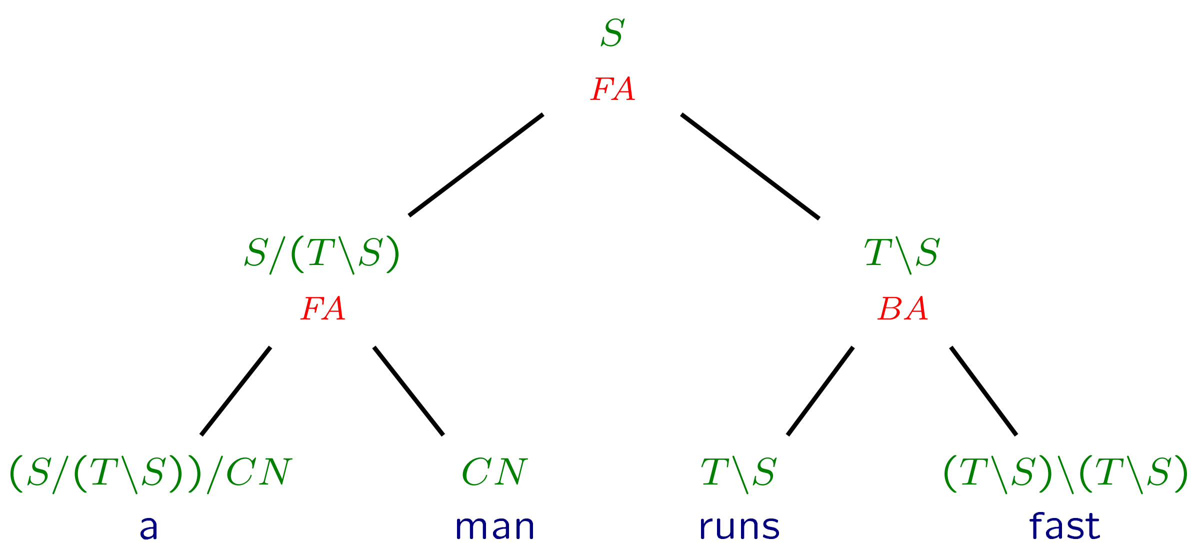
- let $\Sigma$ be a finite vocabulary
- let $\mathcal{B}$ be an enumerable set of basic categories, among which is the axiom $S \in \mathcal{B}$
- the set of categories $Cat(\mathcal{B})$ is the smallest set such that :
  - $\mathcal{B} \subset Cat(\mathcal{B})$
  - $\forall A, B \in Cat(\mathcal{B})$ : $A/B \in Cat(\mathcal{B})$ and $B\backslash A \in Cat(\mathcal{B})$
- a Categorial Grammar $G$ is a finite relation over $\Sigma \times Cat(\mathcal{B})$
- Syntactic rules are expressed by two schemes : $\forall A, B \in Cat(\mathcal{B})$
  - Forward Application $FA$ : $A/B \;\; B \longrightarrow A$
  - Backward Application $BA$ : $B \;\; B\backslash A \longrightarrow A$
- $L(G)$ : set of strings corresponding to a sequence of categories which reduces to $S$

# Categorial Grammars and Recursive Automata

## Definition of a AB-Categorial Grammar

- Let $\mathcal{B} = \{S, T, CN\}$ where $T$ stands for "term" and $CN$ for "common noun"
- $\Sigma = \{$John, runs, a, man, fast$\}$
- $G = \{\langle$John, $T\rangle$, $\langle$runs, $T\backslash S\rangle$, $\langle$a, $(S/(T\backslash S))/CN\rangle$, $\langle$man, $CN\rangle$ $\langle$fast, $(T\backslash S)\backslash(T\backslash S)\rangle\}$

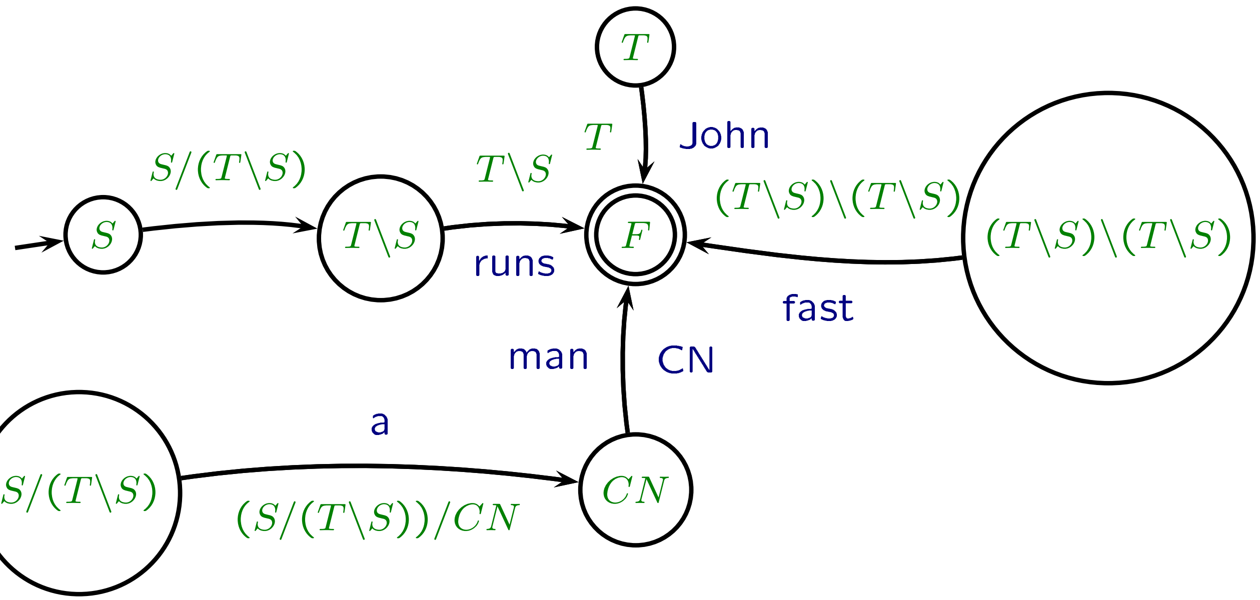


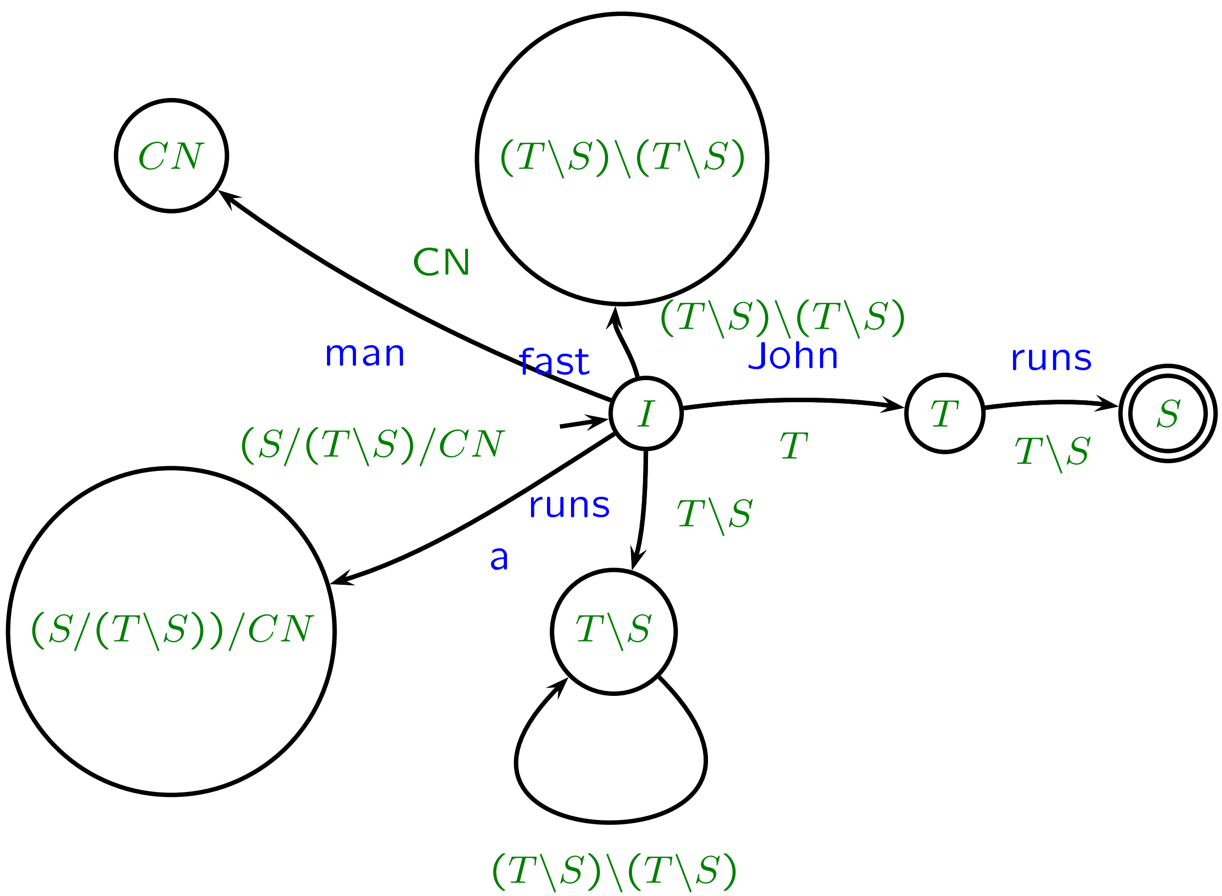



# Categorial Grammars and Recursive Automata

## Definition of Recursive Automata (Tellier06)

- A RA is like a Finite State Automaton except that transitions can be labelled by a state
- Using a transition labelled by a state $Q$ means producing $w \in L(Q)$
- There are two distinct kinds of RA :
  - the $RA_{FA}$-kind where the language $L(Q)$ of a state $Q$ is the set of strings from $Q$ to the final state
  - Every unidirect. $FA$ CG is strongly equivalent with a $RA_{FA}$
  - the $RA_{BA}$-kind where the language $L(Q)$ of a state $Q$ is the set of strings from the initial state to $Q$
  - Every unidirect. $BA$ CG is strongly equivalent with a $RA_{BA}$
- Every CG is equivalent with a pair $MRA = \langle RA_{FA}, RA_{FA} \rangle$


$T$
$S/(T\backslash S)$
$T\backslash S$
$T$
John
$(T\backslash S)\backslash(T\backslash S)$
$S$
$T\backslash S$
$F$
$(T\backslash S)\backslash(T\backslash S)$
runs
fast
man
CN
a
$S/(T\backslash S)$
$CN$
$(S/(T\backslash S))/CN$


$CN$
$(T\backslash S)\backslash(T\backslash S)$
CN
$(T\backslash S)\backslash(T\backslash S)$
man
fast
John
runs
$I$
$T$
$S$
$(S/(T\backslash S)/CN$
$T$
$T\backslash S$
runs
$T\backslash S$
a
$(S/(T\backslash S))/CN$
$T\backslash S$
$(T\backslash S)\backslash(T\backslash S)$

# Outline

1. Introduction
2. Categorial Grammars and Recursive Automata
3. Learning by specialization in both representations
4. Learning from Typed Examples : a new interpretation
5. Conclusion

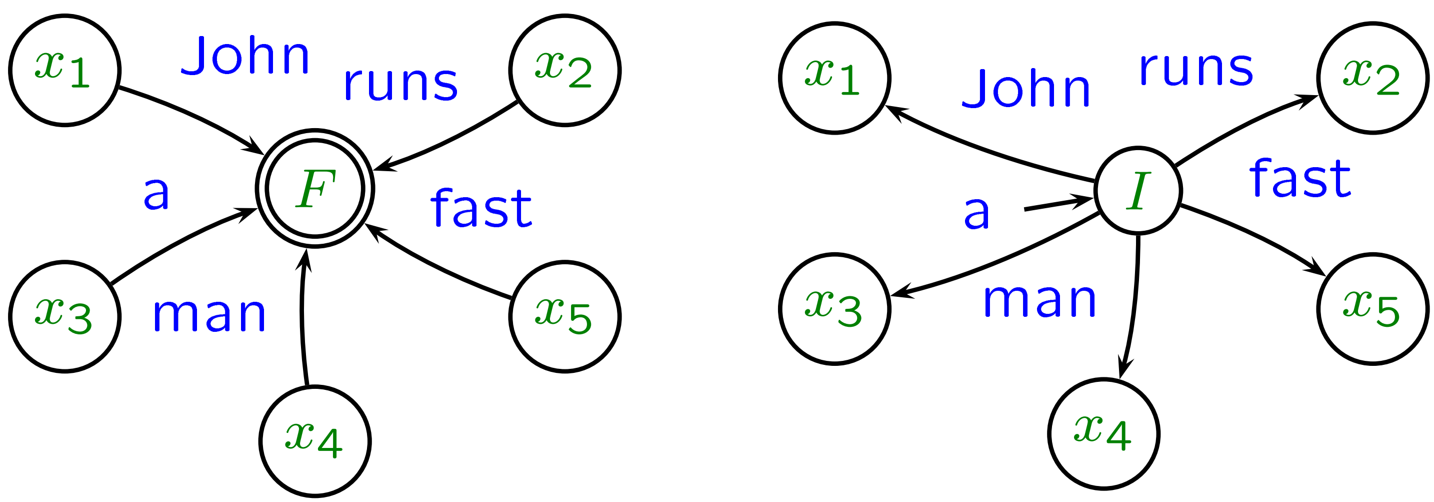
Inference of rigid CGs from strings (Moreau 04)

- Target Class : rigid Categorial Grammars, i.e. at most one category for each word
- Input : a set of sentences
- Learning Algorithm :
  1. associate a distinct unique variable with each word
  2. for each sentence do
     - try to parse the sentence (CYK-like algorithm)
     - induce constraints on the variables
- Output : (disjunctions of) set(s) of constraints, each set corresponding with a (set of) rigid grammar(s)

# Learning by specialization

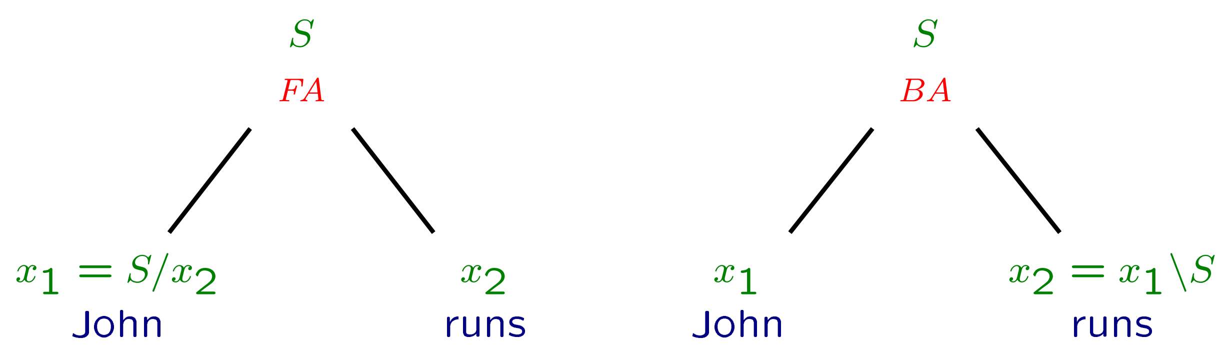
Inference of rigid CGs from strings (Moreau 04) : example

- input data : The set $D = \{$John runs, a man runs fast$\}$
- associate a distinct unique variable with each word :
  $\mathcal{A} = \{\langle$John, $x_1\rangle$, $\langle$runs, $x_2\rangle$, $\langle$a, $x_3\rangle$, $\langle$man, $x_4\rangle$, $\langle$fast, $x_5\rangle\}$
- for every unidirectional CG $G$, there exists a substitution transforming $\mathcal{A}$ into $G$
- $\mathcal{A}$ specifies the set of every unidirectional CGs
- $\mathcal{A}$ can also be represented by a $MRA = \langle RA_{FA}, RA_{BA}\rangle$ :



Inference of rigid CGs from strings (Moreau 04) : example

- the only two possible ways to parse "John runs" :




- to parse "a man runs fast" :
  - theoretically : $5 * 2^3 = 40$ distinct possible ways
  - but some couples of constraints are not compatible with the class of rigid grammars
- main problem with this algo : combinatorial explosion
- to limit it : initial knowledge in the form of known assignments

# Learning by specialization

Effects of constraints on a $MRA = \langle RA_{FA}, RA_{BA} \rangle$

- constraints inferred are of the form :
  - $x_k = x_l$ with $x_k$ and $x_l$ already exist : state and/or transition merges in both the $RA_{FA}$ and the $RA_{BA}$
  - or $x_k = X_m/X_n$ (resp. $x_k = X_m \backslash X_n$) with $X_m, X_n \in Cat(\mathcal{B})$
- the effect of $x_k = X_m/X_n$ (resp. $x_k = X_m \backslash X_n$) in a MRA :
  - $X_m/X_n$ (resp. $x_k = X_m \backslash X_n$) replaces $x_k$ everywhere in the $MRA$
  - every subcategory of $X_m$ and $X_n$ (including themselves) becomes a new state in both the $RA_{FA}$ and the $RA_{BA}$, linked to $F$ (resp. from $I$) by a its name
  - in the $RA_{FA}$ (resp. the $RA_{BA}$), a new transition labelled by $X_m/X_n$ (resp. $X_m \backslash X_n$) links the states $X_m$ and $X_n$
  - the states of the same name are merged
- So : a combination of state splits and state merges
- better founded than the state splits in (Fredouille 00)

# Outline

1. Introduction
2. Categorial Grammars and Recursive Automata
3. Learning by specialization in both representations
4. Learning from Typed Examples : a new interpretation
5. Conclusion

# Learning From Typed Examples

Basic ideas (Dudau, Tellier &Tommasi 01)

- cognitive hypothesis : lexical semantics is learned before syntax
- formalization : words are given with their (Montague's) semantic type
- Types derive from categories by a homomorphism
- Classical example : $h(T) = e,\ h(S) = t,\ h(CN) = \langle e,t\rangle$ and $h(A/B) = h(B\backslash A) = \langle h(B), h(A)\rangle$
- input data : typed sentences are of the form

| John | runs |
|---|---|
| $e$ | $\langle e,t\rangle$ |

| a | man | runs | fast |
|---|---|---|---|
| $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$ | $\langle e,t\rangle$ | $\langle e,t\rangle$ | $\langle\langle e,t\rangle,\langle e,t\rangle\rangle$ |

# Learning From Typed Examples

## Target Class

- The set of CGs such that every distinct category assigned to the same word gives a distinct type
- $\forall \langle v, C_1 \rangle, \langle v, C_2 \rangle \in G,\ C_1 \neq C_2 \Longrightarrow h(C_1) \neq h(C_2)$
- Theorem (Dudau, Tellier & Tommasi 03) : for every CF-language, there exists a grammar $G$ generating it and a morphism $h$ satisfying this condition

## General algorithm (Dudau, Tellier &Tommasi 01)

1. initial set of assignments : introduce variables to represent the class
2. for each sentence
   - try to parse the sentence (CYK-like)
   - induce constraints on the variables
3. Output : (disjunctions) of set(s) of contraint(s), each being represented by a least general grammar

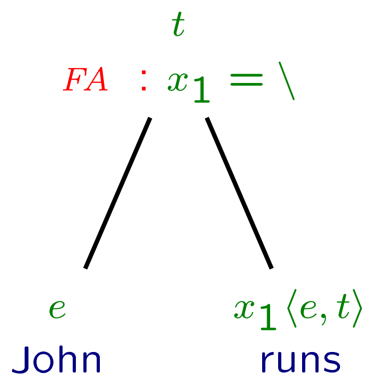
## Example of pre-treatment

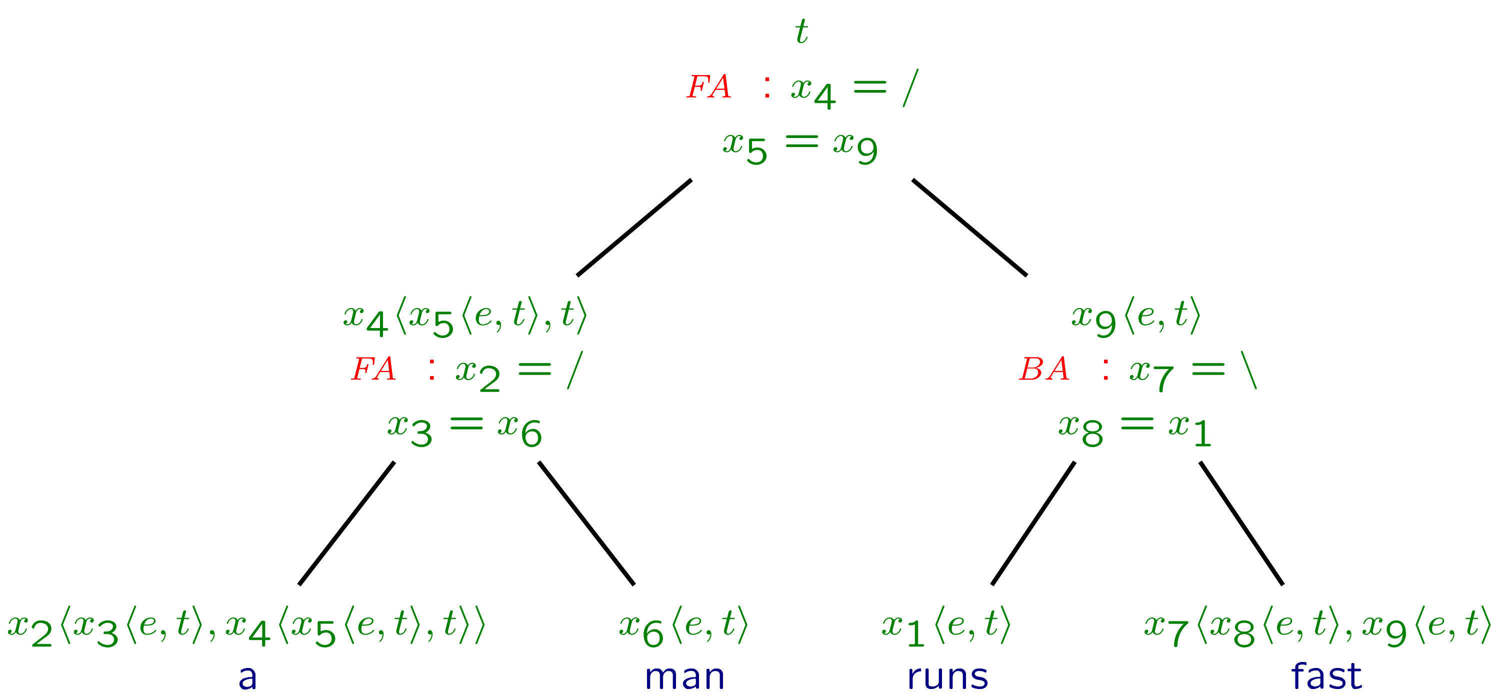
- introduce a distinct variable whose possible values are $/$ or $\backslash$ in front of every subtype
- in our example, the result is of the form :

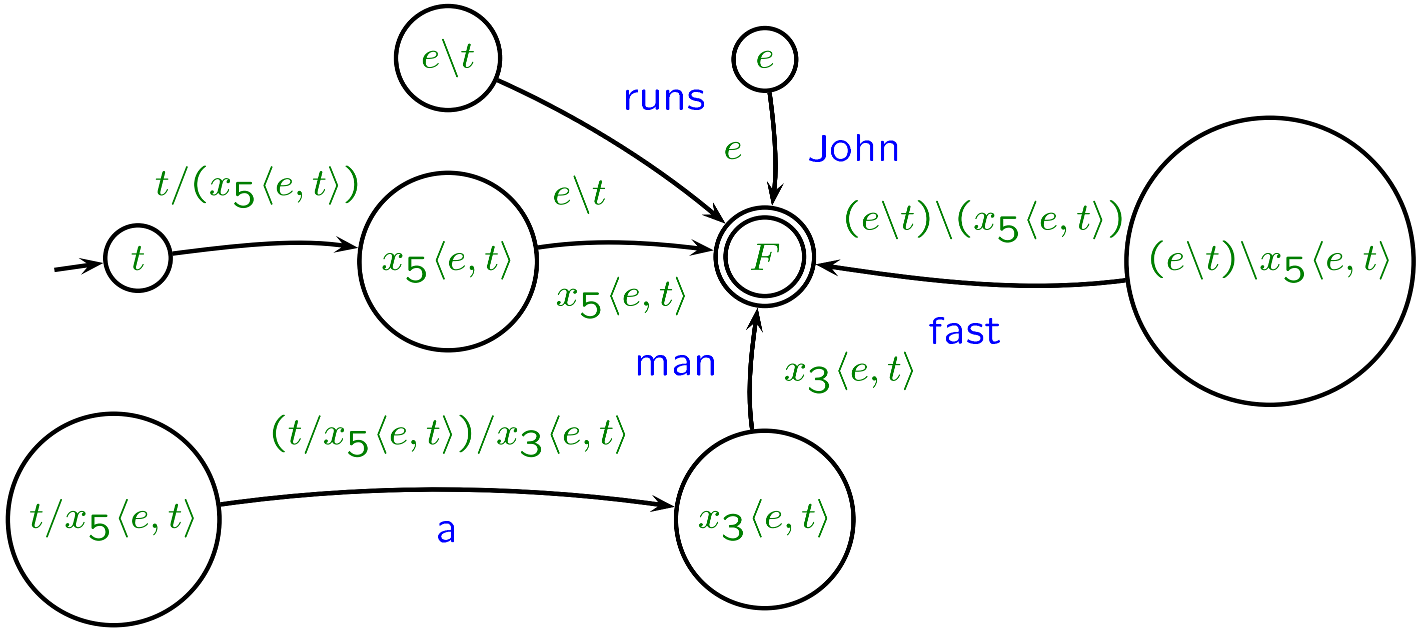
| | |
|---|---|
| John | runs |
| $e$ | $x_1\langle e, t\rangle$ |

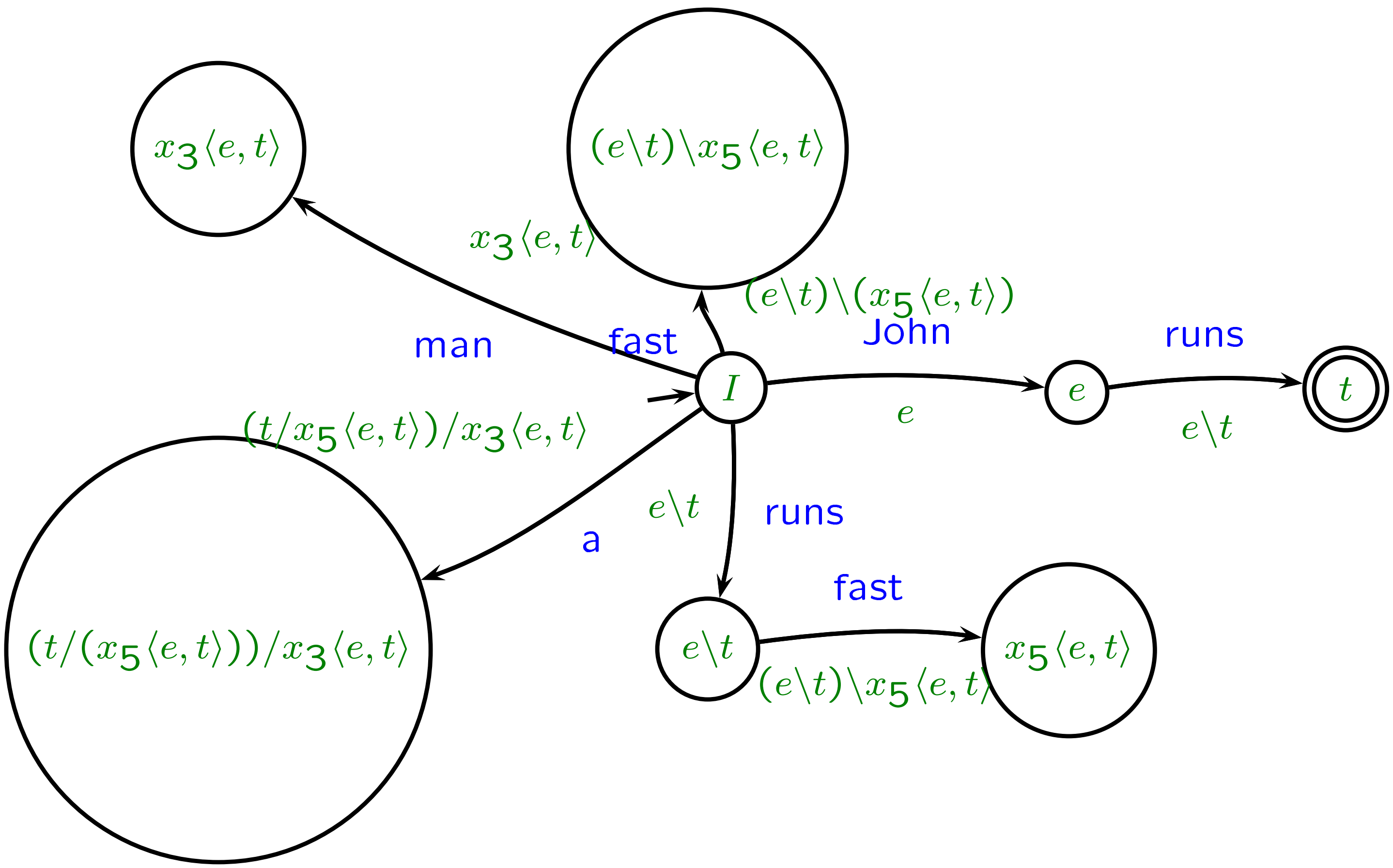
| | | | |
|---|---|---|---|
| a | man | runs | fast |
| $x_2\langle x_3\langle e, t\rangle, x_4\langle x_5\langle e, t\rangle, t\rangle\rangle$ | $x_6\langle e, t\rangle$ | $x_1\langle e, t\rangle$ | $x_7\langle x_8\langle e, t\rangle, x_9\langle e, t\rangle\rangle$ |

## Infering constraints by parsing




$e\backslash t$
$e$
runs
$e$
John
$t/(x_5\langle e,t\rangle)$
$e\backslash t$
$t$
$x_5\langle e,t\rangle$
$F$
$(e\backslash t)\backslash(x_5\langle e,t\rangle)$
$(e\backslash t)\backslash x_5\langle e,t\rangle$
$x_5\langle e,t\rangle$
fast
man
$x_3\langle e,t\rangle$
$(t/x_5\langle e,t\rangle)/x_3\langle e,t\rangle$
$t/x_5\langle e,t\rangle$
$x_3\langle e,t\rangle$
a


$x_3\langle e,t\rangle$
$(e\backslash t)\backslash x_5\langle e,t\rangle$
$x_3\langle e,t\rangle$
$(e\backslash t)\backslash(x_5\langle e,t\rangle)$
John
runs
man
fast
$I$
$e$
$t$
$e$
$e\backslash t$
$(t/x_5\langle e,t\rangle)/x_3\langle e,t\rangle$
$e\backslash t$
runs
a
fast
$(t/(x_5\langle e,t\rangle))/x_3\langle e,t\rangle$
$e\backslash t$
$x_5\langle e,t\rangle$
$(e\backslash t)\backslash x_5\langle e,t\rangle$

## Sum-up

- mix of state splits and state merges
- Types contain in themselves where splits are possible
- not every (complex) state can be merged : states are typed in the sense of (Coste & alii 2004)
- the use of types reduces the combinatorial explosion of possible splits
- types helph to converge to the correct solution quicker

# Learning From Typed Examples

## Sum-up

| vocabulary | Moreau's initial assigment | target category | pre-treated type |
|---|---|---|---|
| a | $x_1$ | $(S/(T\backslash S))/CN$ | $x_2\langle x_3\langle e,t\rangle, x_4\langle x_5\langle e,t\rangle, t\rangle\rangle$ |
| man | $x_2$ | $CN$ | $x_6\langle e,t\rangle$ |
| runs | $x_3$ | $T\backslash S$ | $x_1\langle e,t\rangle$ |

- there exists a substitution, thus a homomorphism between Moreau's assignments and categories
- there exists a homomorphism between categories and types (Principle of compositionality)
- the starting point is either a lower bound or an upper bound
- the "good substitution" is well constrained by types

# Outline

1. Introduction
2. Categorial Grammars and Recursive Automata
3. Learning by specialization in both representations
4. Learning from Typed Examples : a new interpretation
5. Conclusion

# Conclusion

Main contributions

- we mainly propose a new perspective on already known algorithms
- the correspondance between Categorial Grammars and recursive automata is fruitful
- MRA can represent sets of grammars corresponding to search spaces
- specialization strategies require additional knowledge (like semantic types)
- natural language is probably learnt by specialization by children
- specialization techniques deserve further investigation (better for incrementality...)