GRAMMATICAL INFERENCE BY SPECIALIZATION AS A STATE SPLITTING STRATEGY

ISABELLE TELLIER

LIFL-Inria
university of Lille
isabelle.tellier@univ-lille3.fr

We exhibit connexions beteween two already known learning algorithms developped in different backgrounds. This allows to show that learning classical (or AB) categorial grammars by specialization can be identified with a "state splitting" strategy, in a search space made of extended automata. It also leads to a new interpretation of why it is possible to learn categorial grammars from semantically typed (in Montague's sense) examples.

In recent papers (Tellier 2005; Tellier 2006), it was shown that classical (or AB) categorial grammars (CGs in the following) could easily be represented by extended automata called recursive automata (RA). This translation allowed to exhibit connexions between two previously distinct approaches of grammar learning from positive examples: the one used in the "BP algorithm" to learn subclasses of CGs (Buszkowski and Penn 1990; Kanazawa 1996; Kanazawa 1998) and the one used to learn regular grammars represented by finite state automata (Angluin 1982; Dupont et al. 1994). In particular, the "state merging" operator used in automata learning was shown to be nothing but a special case of the "unification of variables" operator used in the BP algorithm.

The previous learning algorithms all belung to the family of *generalization strategies*. A generalization strategy applies as follows: the initial hypothesis is a "least general grammar" representing the available examples. Then, an operator is used to generalize this hypothesis until it belongs to the target class. But in symbolic machine learning in general, and in grammatical inference in particular, there also exists a lesser known family of *specialization strategies*. In such strategies, the initial hypothesis is the whole target class of grammars. Each example is considered as a *constraint* which restricts this space, until the space is reduced to a single grammar.

1. Introduction

The aim of this paper is to show that the translation of CGs into RA, which has helped to better understand the family of generalization strategies, can also help to better understand the family of specialization strategies. As a matter of fact, as it was

the case for generalization strategies, specialization approaches have been proposed independantly in two distinct backgrounds: to learn CGs in the one hand, and to learn regular grammars represented by finite state automata in the other hand.

To reach this aim, we first need to briefly recall in section 2. how to transform a CG into a recursive automaton. For sake of simplicity, we restrict ourselves in most of this article to unidirectional CGs, but the definitions can be extended to plain CGs. In section 3., we first present the specialization strategy described in (Moreau 2004), allowing to learn rigid CGs from positive examples. We then explain how it relates to another specialization strategy, which targets regular languages represented by finite state automata (Fredouille and Miclet 2000). As expected, we show that Moreau's algorithm can be interpreted as a "state splitting" strategy applying on RA. Finaly, the whole picture is completed in section 4. by a new interpretation of yet another already known algorithm allowing to learn CGs from semantically typed (in Montague's sense) examples (Dudau-Sofronie et al. 2001). It appears to be an efficiently controled specialization approach.

2. From categorial grammars to recursive automata

2.1. Basic definitions of categorial grammars

Definition 1 (unidirectional classical categorial grammars) *Let $\mathcal{B}$ be an enumerable set of **basic categories** containing the **axiom** $S \in \mathcal{B}$. $Cat(\mathcal{B})$ is the smallest set such that $\mathcal{B} \subseteq Cat(\mathcal{B})$ and for any $A, B \in Cat(\mathcal{B})$, $A/B \in Cat(\mathcal{B})$ (for bidirectional CGs, we also have $B\backslash A \in Cat(\mathcal{B})$). For every finite **vocabulary** Σ, a **unidirectional categorial grammar** G is a finite relation over $\Sigma \times Cat(\mathcal{B})$. We note $\langle w, C\rangle \in G$ the assignment of the category $C \in Cat(\mathcal{B})$ to the word $w \in \Sigma$. In classical (or AB) unidirectional categorial grammars (UCGs in the following) the only syntactic rule, called Forward Application and noted FA is: $\forall A, B \in Cat(\mathcal{B})$, $A/B \;\; B \rightarrow A$. The language of G is: $L(G)$=$\{w = v_1 \dots v_n \in \Sigma^+ \mid \forall i \in \{1, \dots, n\}, \exists A_i \in Cat(\mathcal{B})$ such that $\langle v_i, A_i\rangle \in G$ and $A_1 \dots A_n \rightarrow^* S\}$, where $\rightarrow^*$ is the reflexive and transitive closure of the relation $\rightarrow$.*

Let for example, $\mathcal{B} = \{S, CN, IV\}$ (where CN stands for "common noun" and IV for "intransitive verbs"), $\Sigma = \{John,\ runs,\ a,\ man\}$ and $G = \{\langle John, S/IV\rangle$, $\langle runs, IV\rangle, \langle a, (S/IV)/CN\rangle, \langle man, CN\rangle\}$. This over-simple UCG only recognizes the sentences "John runs" and "a man runs".

2.2. Recursive automata and their language

Definition 2 (recursive automaton) *A **recursive automaton** R is a 5-tuple $R = \langle Q, \Sigma, \gamma, q_0, F\rangle$ such that Q is the finite set of **states** of R, Σ is its finite **vocabulary**, $q_0 \in Q$ its (unique) **initial state** and $F \in Q$ its (unique) **final state**. γ is the **transition function** of R, defined from $Q \times (\Sigma \cup Q)$ to 2^Q.*

We restrict ourselves to recursive automata (RA in the following) with unique initial and final states, but it is not a crucial choice The only important difference between this definition and the classical definition of finite state automata is that in a RA, it is possible to *label a transition either by an element of* Σ *or by an element of* Q. To use a transition labeled by a state $q \in Q$, you need to generate a string belonging to the language $L_R(q)$ of this state q, i.e. a string corresponding to a path starting at the state q and reaching the final state F. The general definition of the set $\{L_R(q)|q \in Q\}$ is thus recursive: when it exists, it is a smallest fix-point. In fact, RA are a special case of Recursive Transition Networks (Woods 1970).

Definition 3 (language of a RA) *We define the set of languages* $L_R(q)$*, for every* $q \in Q$ *as the smallest set satisfying: (i)* $L_R(F) = \epsilon$*; (ii) if there exists a transition labeled by* $a \in \Sigma$ *between* q *and* q'*, i.e.* $q' \in \gamma(q, a)$ *then:* $a.L_R(q') \subseteq L_R(q)$*; (iii) if there exists a transition labeled by* $r \in Q$ *between* q *and* q'*, i.e.* $q' \in \gamma(q, r)$ *then:* $L_R(r).L_R(q') \subseteq L_R(q)$*. Finaly, the language of* R *is:* $L(R) = L_R(q_0)$*.*

2.3. From unidirectional CGs to RA

Every UCG can be transformed into a strongly equivalent RA, i.e. a RA generating the same structural descriptions (Tellier 2006). Let G be a UCG over $\Sigma \times Cat(\mathcal{B})$. Build $R = \langle Q, \Sigma, \gamma, q_0, F\rangle$ as follows:

- let N be the set of every subcategory of a category assigned to an element of Σ in G (a category is a subcategory of itself). Then $Q = N \cup \{F\}$ with $F \notin N$. The initial state $q_0 = S$, the final one is F.

- For every $q \in Q$, define a transition labeled by q between the state q and F (that is, $F \in \gamma(q, q)$). For every $A/B \in N$, define a transition labelled by A/B between the states A and B ($B \in \gamma(A, A/B)$). For every $\langle w, C\rangle \in G$, add a transition labelled by w between the states C and F ($F \in \gamma(C, w)$).

The example UCG of subsection 2.1. can be transformed into the RA of Figure 1. (after an easy simplification process has been applied for readability):

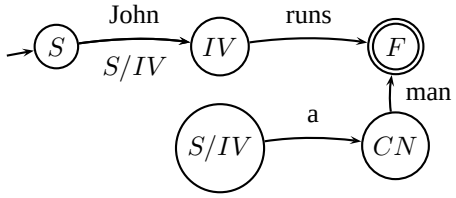

Figure 1: RA equivalent with a UCG

3. Learning by specialization

3.1. Learning rigid UCG from positive examples

A rigid CG G is a CG in which every word $w \in \Sigma$ is assigned at most one category C. Kanazawa has proved (Kanazawa 1998) that the set of every (bidirectional) CG is learnable "in the limit" (i.e. in the sense of Gold 1967) from positive examples, i.e. from sentences. Two distinct learning algorithms are now available for this purpose: Kanazawa's, derived from "BP" (Buszkowski and Penn 1990), is a *generalization strategy*; the one proposed in (Moreau 2004) is a *specialization strategy*. It is the latter, in its unidirectional version, that we will briefly recall here.

At its start, each member of the vocabulary used at least once in the available example sentences is assigned a distinct variable. For example, for the sentences {"John runs", "a man runs"}, the initial assignment is $\mathcal{A} = \{\langle John, x_1\rangle, \langle runs, x_2\rangle, \langle a, x_3\rangle, \langle man, x_4\rangle\}$. $\mathcal{A}$ specifies *a set of grammars*: the set of CGs G such that there exists an substitution h satisfying $h(\mathcal{A}) = \{\langle w, h(C)\rangle | \langle w, C\rangle \in \mathcal{A}\} \subseteq G$. It is obvious that the inital set $\mathcal{A}$ always specifies the whole set of rigid CGs built on Σ.

Then, each sentence is parsed with the assigments in $\mathcal{A}$. The only possible way to parse "John runs" with only *FA* is to substitute S/x_2 to x_1. This substitution is a *constraint* that the variable x_1 must satisfy: x_1 is thus replaced by S/x_2 in $\mathcal{A}$. To parse "a man runs", two solutions are possible: either $x_3 = (S/x_2)/x_4$, either $x_3 = S/x_5$ and $x_4 = x_5/x_2$ (with x_5 a new variable). $\mathcal{A}$ then becomes a disjunction of distinct possible sets of assignments. A combinatorial explosion can occur.

3.2. State merges and state splits

The previous algorithm can now be interpreted in terms of operations applying on RA. As we have seen, $\mathcal{A}$ is a disjunction of sets of assignments. Each of these sets can be transformed into a RA, as described in section 2.3.. What is the effect of a constraint on a RA ?

For UCG, the constraints always take the form: $x_k = x_l$, with x_k and x_l already introduced variables, or $x_k = X_m/X_n$, with X_m and X_n any *category built on the set of every variables union S*. The effect of a constraint of the form $x_k = x_l$ on a RA is a *state merge*. The effect of a constraint of the form $x_k = X_m/X_n$ can be decomposed in three steps: (i) X_m/X_n replaces x_k everywhere in the RA, (ii) every subcategory of X_m and X_n (including themselves) becomes a new state, linked to the state F by a transition labeled by its name, and every / inside X_m/X_n becomes a transition, labeled by the fraction of the names of the linked states (at least, X_m and X_n are linked by X_m/X_n), (iii) the states of the same name are merged.

This operation can be compared to the "state splitting strategy" proposed in (Fredouille and Miclet 2000) to learn finite state automata by specialization. For example, the constraint $x_1 = S/x_2$ has the effect of splitting the state x_1 into two new states: S and x_2 (then, as a state named x_2 already exists, the new one is merged with the previous one). But our specialization operation is more general, because of

the recursive nature of the automata on which it applies. It is also better founded, because it is the formal counterpart of well-defined substitutions.

4. Learning from typed examples

The idea of learning CGs from typed (in Montague's sense) examples was introduced in (Dudau-Sofronie et al. 2001). Montague's types are derived from categories by a morphism, and associated with the vocabulary in the sentences. They can be interpreted as semantic information available in the environment or previously learned. We illustrate the learning strategy and its effects on RA on a simple example. Let:

a	man	runs
$\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$	$\langle e,t\rangle$	$\langle e,t\rangle$
$(tx_1(tx_2e))x_3(tx_4e)$	tx_5e	tx_6e

In this typed sentence, t and e are the usual montagovian basic types. The last line is the result of a simple pre-treatment to reorder the types in a UCG fashion and to introduce distinct variables at every place where an operator / could occur. When we transform this initial assignment into a RA, we obtain three states (plus "F"), each linked to the unique final state F, with their own name and "a", "man" and "runs" as respective labels. The learning algorithm applies as in section 3.1.: it consists in trying to parse the sentence, by defining constraints on the variables.

In a first step, the only way to apply FA between two consecutive types of the example is to define: $x_3 = /$. This, as already seen, provokes a state split. But the FA rule relying on the introduced operator can apply only if $tx_4e = tx_5e$, that is if $x_4 = x_5$, which specifies a state merge. Not every couple of states can be merged at this step: states are also semantically typed in the sense of (Coste et al. 2004). The result of this step is the set: $\{\langle a, (tx_1(tx_2e))/(tx_4e)\rangle, \langle man, tx_4e\rangle, \langle runs, tx_6e\rangle\}$, corresponding with the first (simplified) RA of Figure 2.

In a second step, similar to the first one, we have: $x_1 = /$ and $x_2 = x_6$. The type t, corresponding with the category S, is now a subcategory of an assigned category, and a new state t playing the role of *initial state* is thus introduced. The RA obtained, on the right in Figure 2., is isomorph to the one of Figure 1., and recognizes the initial sentence. The types helped to cenverge to the correct solution quicker.

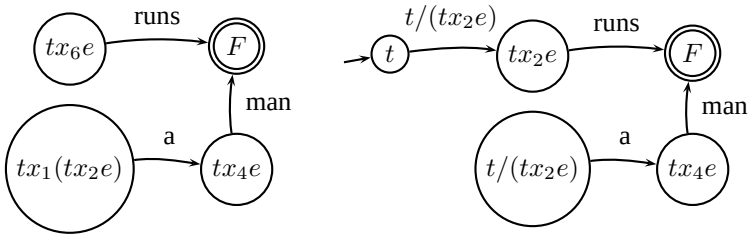

Figure 2: RA obtained when learning from typed examples

5. Conclusion

In this paper, we propose a new perspective on already known techniques. First, we see that RA are able to represent the *search space* of a specialization learning algorithm. In fact, as $\mathcal{A}$ is a disjunction of sets, the search space of Moreau's algorithm can be represented by *a disjunction of RA*. Second, we show that the algorithm to learn CGs from typed examples proposed in (Dudau-Sofronie et al. 2001) is a specialization strategy with typed constraints. The initial semantic types associated with the elements of the vocabulary specify some kind of maximal bound on the possible splits to be performed, allowing to limit the combinatorial explosion of solutions.

Bibliography

Angluin, D.: 1982, Inference of reversible languages, *J. ACM* 29(3), 741–765

Buszkowski, W. and Penn, G.: 1990, Categorial grammars determined from linguistic data by unification, *Studia Logica* 49, 431–454

Coste, F., Fredouille, D., Kermovant, C., and de la Higuera, C.: 2004, Introducing domain and typing bias in automata inference, in *proceedings of the 7th ICGI*, Vol. 3264 of *LNAI*, pp 115–126, Springer Verlag

Dudau-Sofronie, D., Tellier, I., and Tommasi, M.: 2001, Learning categorial grammars from semantic types, in *proceedings of the 13th Amsterdam Colloquium*, pp 79–84

Dupont, P., Miclet, L., and Vidal, E.: 1994, What is the search space of the regular inference, in *ICGI'94*, Vol. 862 - Grammatical Inference and Applications of *LNAI*, pp 25–37, Springer Verlag

Fredouille, D. and Miclet, L.: 2000, Experiences sur l'inference de langage par specialisation, in *proceedings of CAP'2000*, pp 117–130

Gold, E.: 1967, Language identification in the limit, *Inform. Control* 10, 447–474

Kanazawa, M.: 1996, Identification in the limit of categorial grammars, *Journal of Logic, Language and Information* 5(2), 115–155

Kanazawa, M.: 1998, *Learnable Classes of Categorial Grammars*, The European Association for Logic, Language and Information, CLSI Publications

Moreau, E.: 2004, Apprentissage partiel de grammaires lexicalises, *TAL* 45(3), 71–102

Tellier, I.: 2005, When categorial grammars meet regular grammatical inference, in *proceedings of the 5th LACL*, Vol. 4492 of *LNAI*, pp p.317–332, Springer Verlag

Tellier, I.: 2006, Learning recursive automata from positive examples, *Revue d'Intelligence Artificielle* New Methods in Machine Learning(20/2006), 775–804

Woods, W. A.: 1970, Transition network grammars of natural language analysis, *Communications of the ACM* (13), 591–606