The Computational Complexity of Taxonomic Inference

Radford Neal

December, 1989

McAllester, Given, and Fatima [10] have developed a procedure for infering taxonomic relationships between classes defined by predicates and relations. Their decision procedure runs in $O(n^3)$ time and $O(n^2)$ space on a sequential random-access machine (RAM). I have investigated the computational complexity of this inference problem with a view to seeing whether faster sequential algorithms or good parallel algorithms might be found.

A restricted form of the taxonomic inference task of McAllester, *et al*, which I will call *taxonomic closure*, can be seen as a generalization of *congruence closure*, which has been investigated by Kozen [6], Nelson and Oppen [11], and Downey, Sethi, and Tarjan [4]. I will show that the decision problems corresponding to both taxonomic closure and congruence closure are P-complete, even if terms are restricted to contain only monadic function applications. Thus these problems probably cannot be efficiently parallelized. I also show that the monadic taxonomic closure decision problem is complete for two-way non-deterministic pushdown automata (2NPDA) — a problem class for which the best known algorithm takes $O(n^3)$ time on a sequential RAM.

The negative implications of these results for taxonomic and congruence closure apply to the more powerful taxonomic inference system of McAllester, *et al*, and to the later extension of this system to "Montague literals" by McAllester and Givan [9]. I will discuss the significance of these results for engineering applications and for McAllester and Givan's speculation that the decision procedure for Montague literals might explain some aspects of natural language.

A Taxonomic Inference System

The system for taxonomic inference that I will discuss is a subset of a more general system of inference for "taxonomic literals" described by McAllester, *et* *al* [10]. Their system permits statements that one set is included in another, that two sets have a non-null intersection, that a set is non-empty, and that a set contains no more than one element, as well as the negations of such statements. The system I will describe allows only positive assertions that one set is included in another.

This taxonomic inference system deals with a set of *predicates* on one argument, and a set of *relations* between two or more arguments. Rather than apply these predicates and relations to individual items, a predicate symbol is used to represent the set of items for which the predicate is true, and the symbol for a relation between n items is used to represent the function that takes $n-1$ sets as arguments and returns the set of all items that are so related to some items taken from the argument sets. Sentences in the system assert that one set defined in this fashion is contained in another such set.

For example, given the predicates *Poor*, *Rock*, and *Hardplace*, and the relation *Between*, the sentence

$$Poor \subseteq Between(Rock, Hardplace)$$

asserts that all the poor are between a rock and a hardplace, or in standard predicate calculus

$$(\forall x)\ Poor(x) \ \Rightarrow\ (\exists u, v)\ Rock(u)\ \&\ Hardplace(v)\ \&\ Between(u, v, x)$$

More formally, a *term* is either one of a set of predicate symbols, P_i, or it is one of a set of relation symbols, R_i, say of arity n, applied to the terms $\alpha_1, \ldots, \alpha_{n-1}$ as arguments, written as $R_i(\alpha_1, \ldots, \alpha_{n-1})$. A *sentence* is of the form $\alpha \subseteq \beta$, where α and β are terms.

An *interpretation* consists of a domain, D, an assignment of a set $\hat{P}_i \subseteq D$ to each predicate symbol P_i, and an assignment of a relation $\hat{R}_i \subseteq D^n$ to each n-ary relation symbol, R_i. Given an interpretation, each term α is assigned a subset of the domain by the function $I(\alpha)$, defined recursively as follows:

- $I(P_i) = \hat{P}_i$
- $I(R_i(\alpha_1, \ldots, \alpha_{n-1})) = \{\ x \mid (\exists u_1, \ldots, u_{n-1})$
  $u_1 \in I(\alpha_1)\ \&\ \cdots\ \&\ u_{n-1} \in I(\alpha_{n-1})\ \&\ \hat{R}_i(u_1, \ldots, u_{n-1}, x)\ \}$

A sentence $\alpha \subseteq \beta$ is *true* under an interpretation if and only if $I(\alpha) \subseteq I(\beta)$. A set of sentences Σ *entails* a sentence ξ, written $\Sigma \models \xi$, if and only if for all interpretations under which the sentences of Σ are true, the sentence ξ is true as well.

We can *derive* sentences from other sentences via the following *inference rules*:

1) $\alpha_1 \subseteq \beta_1,\ \alpha_2 \subseteq \beta_2,\ \cdots\ \rightarrow\ R(\alpha_1, \alpha_2, \ldots) \subseteq R(\beta_1, \beta_2, \ldots)$

2) $\alpha \subseteq \beta,\ \beta \subseteq \gamma\ \rightarrow\ \alpha \subseteq \gamma$

3) $\rightarrow\ \alpha \subseteq \alpha$

Here, α, β, γ, α_1, β_1, etc. represent any terms, and R is any relation symbol of the appropriate arity. Inference rule (1) says that if one set is included in another, then the set of things related to the first set is included in the set of things related to the second. For example, from the fact that all surgeons are doctors, one can infer that all children of surgeons are children of doctors. Rules (2) and (3) simply state that set inclusion is transitive and reflexive.

Define $\Sigma \vdash \xi$ to mean that the sentence ξ can be derived from the sentences in Σ by applying the above rules of inference — i.e. there is a sequence of sentences (a *proof*), $\zeta_1, \zeta_2, \ldots, \zeta_m = \xi$, such that each ζ_i is either in Σ or follows from earlier sentences via some rule of inference, $\zeta_{p_1}, \zeta_{p_2}, \ldots \rightarrow \zeta_i$, with $p_1, p_2, \ldots < i$.

These inference rules are semantically *sound* and *complete* with respect to the entailment relation:

Theorem 1 *For any set of sentences Σ and any sentence ξ, $\Sigma \models \xi$ if and only if $\Sigma \vdash \xi$.*

This theorem is proved in the appendix. McAllester, *et al* prove the refutation completeness of their more general system, from which the full completeness of this system can also be deduced.

This inference system can be given semantics unrelated to that of taxomomic relations. For example, "predicates" might be real numbers, "relations" be monotonically non-decreasing real functions, and "$\subseteq$" be "$\leq$" applied to real numbers.

A Fundamental Theorem

A decision procedure for the taxonomic inference system developed here is made possible by a theorem limiting the class of sentences that need be considered when searching for a proof. This theorem has an analogue in the system of McAllester, *et al*, where it is proved via a semantic argument. Here, I will give a syntactic proof of the theorem for the restricted system.

This fundamental theorem states that if a sentence can be proved from a set of premises, then there is a proof in which only terms that are *relevant* to the conclusion or premises occur. A term is relevant to a set of sentences, Δ, if it occurs as a term or sub-term in a sentence in Δ. For example, the terms relevant to $\{\ R(P,Q),\ R(R(P,P),P)\ \}$ are P, Q, $R(P,Q)$, $R(P,P)$, and $R(R(P,P),P)$.

Theorem 2 *For any set of sentences Σ and any sentence ξ, if $\Sigma \vdash \xi$ then there is a proof of ξ in which only terms that are relevant to $\Sigma \cup \{\xi\}$ occur.*

Proof By duplicating or removing sentences, any proof can be converted to a "tree-like" proof in which each sentence (other than the conclusion) is used to justify exactly one later sentence. I will show how any tree-like proof of $\Sigma \vdash \xi$ that contains terms not relevant to $\Sigma \cup \{\xi\}$ can be transformed into another tree-like proof with fewer occurences of the largest irrelevant term, while adding only smaller irrelevant terms. Repeated application of this procedure must progressively eliminate all irrelevant terms, in order of diminishing size.

Let $\alpha_0 \subseteq \beta_0$ be the last sentence in the tree-like proof of $\Sigma \vdash \xi$ that is derived from premises containing the largest irrelevant term, and note that α_0 and β_0 do not themselves contain this term. This sentence cannot be derived via rule (3), since that inference rule has no premises. Neither can it be derived via rule (1), since the conclusion of that rule contains terms that are larger, and just as irrelevant, as those in its premises. Therefore, $\alpha_0 \subseteq \beta_0$ must be derived from $\alpha_0 \subseteq \delta$ and $\delta \subseteq \beta_0$ via rule (2), with δ being the largest irrelevant term in the proof.

Following the chains of inference backward, $\alpha_0 \subseteq \delta$ and $\delta \subseteq \beta_0$ will have been derived by chains of zero or more applications of rule (2), of the form

$$\alpha_0 \subseteq \alpha_1 \subseteq \cdots \subseteq \alpha_n \subseteq \delta, \quad \delta \subseteq \beta_m \subseteq \cdots \subseteq \beta_1 \subseteq \beta_0$$

The sentences $\alpha_n \subseteq \delta$ and $\delta \subseteq \beta_m$ cannot be in Σ, since δ is an irrelevant term. If either of these sentences is derived via rule (3), it is redundant and can be removed from the chain. Since applications of rule (2) have already been accounted for, this leaves both $\alpha_n \subseteq \delta$ and $\delta \subseteq \beta_m$ as consequences of rule (1), which means that α_n, δ, and β_m must have the forms $R(\bar{\alpha}_1, \bar{\alpha}_2, \ldots)$, $R(\bar{\delta}_1, \bar{\delta}_2, \ldots)$, and $R(\bar{\beta}_1, \bar{\beta}_2, \ldots)$ respectively, and the sentences $\bar{\alpha}_1 \subseteq \bar{\delta}_1$, $\bar{\delta}_1 \subseteq \bar{\beta}_1$, $\bar{\alpha}_2 \subseteq \bar{\delta}_2$, $\bar{\delta}_2 \subseteq \bar{\beta}_2$, etc. must appear earlier in the proof.

A tree-like proof of $\Sigma \vdash \xi$ without the occurences of δ can now be obtained. The sentence $\bar{\alpha}_1 \subseteq \bar{\beta}_1$ can be introduced, justified by rule (2) applied to $\bar{\alpha}_1 \subseteq \bar{\delta}_1$ and $\bar{\delta}_1 \subseteq \bar{\beta}_1$. Similarly, introduce $\bar{\alpha}_2 \subseteq \bar{\beta}_2$, etc. These allow the sentence $R(\bar{\alpha}_1, \bar{\alpha}_2, \ldots) \subseteq R(\bar{\beta}_1, \bar{\beta}_2, \ldots)$, which is the same thing as $\alpha_n \subseteq \beta_m$, to be introduced via rule (1). The sentence $\alpha_0 \subseteq \beta_0$ can now be derived by the chain

$$\alpha_0 \subseteq \alpha_1 \subseteq \cdots \subseteq \alpha_n \subseteq \beta_m \subseteq \cdots \subseteq \beta_1 \subseteq \beta_0$$

Finally, we delete the sentences $\alpha_n \subseteq \delta$ and $\delta \subseteq \beta_m$ — they are no longer needed here, and neither are they required elsewhere, since the previous proof was tree-like. This deletion makes the new proof tree-like as well.

The new proof has two less occurences of δ, which was the largest irrelevant term in the old proof. New occurences of the $\bar{\alpha}_i$ and $\bar{\beta}_i$ have been introduced, but if these are irrelevant, they must be smaller than δ. Repeated application of this procedure will therefore eliminate first all the largest irrelevent terms, then all the next largest, etc. until all terms not relevant to $\Sigma \cup \{\xi\}$ have been eliminated. ■

This theorem is applied in the next section.

The Taxonomic Closure Problem

McAllester, *et al* give a decision procedure for their "taxonomic literals" that runs in $O(N^3)$ time on a sequential RAM. The less expressive system described here likewise has a cubic time decision procedure, and like that of McAllester, *et al* it involves finding the closure of a set of axioms under the inference rules.

The closure problem can be formulated as follows. Given a set, Σ, of axiomatic sentences, and a set, Q, of query terms, the *taxonomic closure* of Σ with Q is the set of all sentences that can be derived from Σ via the taxonomic inference rules using only terms that occur as terms or sub-terms in Σ or Q. By theorem (2), this is also the set of all sentences built from such terms that can be derived from Σ.

Theorem 3 *The taxonomic closure problem can be solved in $O(n^3)$ time and $O(n^2)$ space on a sequential RAM, where n is the length of the input (an encoding of the axioms and query terms)*[1].

Proof The following algorithm finds the taxonomic closure of Σ with Q:

1) Create a table, T, that for each sentence built from terms that occur in Σ or Q records whether that sentence has been derived from Σ. Initialize

[1] The length of the input can be taken to be proportional to the number of predicate and relation symbols, in which case the $O(n^3)$ time bound applies when elementary operations are assumed to take constant time. Alternatively, the encoding of a symbol can be assumed to require $\log n$ space, in which case the theorem holds when elementary operations take time logarithmic in the operand size.

all entries in T to false.

2) Set the entry in T to true for each sentence in Σ and for each sentence that can be derived via rule (3). Whenever an entry for a sentence ζ is set to true when it was previously false, find all sentences that can then be derived via inference rules for which ζ is a premise, and set their entries in T to true as well, recursively applying this condition.

3) When step (2) terminates, the table T represents the taxonomic closure of Σ with Q.

$O(n)$ terms are relevent to Σ or occur in Q. These terms form $O(n^2)$ sentences, and hence this algorithm changes at most $O(n^2)$ entries in T from false to true. For each such change, a search for inferences that become possible must be made. $O(n)$ time suffices to find all new inferences via rule (2), contributing at most $O(n^3)$ to the total time for the algorithm. Inferences via rule (1) can be detected by maintaining a count of premises remaining to be derived for each possible conclusion, decrementing these counts as premises are found to be true, and noticing when the count reaches zero. The sum of the initial count values can be at most $O(n^2)$, which limits the time spent in this operation. The total time required is thus $O(n^3)$, while the space required is just the $O(n^2)$ needed for T, the up to $O(n^2)$ for the counts, and $O(n^2)$ for various list structures needed to implement the above operations in the requisite time. ■

The *taxonomic closure decision problem* asks whether a sentence $\alpha \subseteq \beta$ is in the taxonomic closure of Σ with $\{\alpha, \beta\}$ — i.e. whether $\alpha \subseteq \beta$ can be derived from the sentences of Σ. This problem can be solved by first computing the full taxonomic closure and then checking whether it contains $\alpha \subseteq \beta$. I know of no method that is substantially better than this in the worst case.

The Congruence Closure Problem

The *congruence closure problem*, studied by Kozen [6], Nelson and Oppen [11], and Downey, Sethi, and Tarjan [4], is the analogue of the taxonomic closure problem with an equality relation ($=$) rather than a partial order ($\subseteq$). The syntax of terms is identical, but the symbols are typically seen as representing arbitrary constants and functions rather than predicates and relations.

The inference rules used for congruence closure are the following:

1) $\alpha_1 = \beta_1,\ \alpha_2 = \beta_2,\ \cdots\ \rightarrow\ R(\alpha_1, \alpha_2, \ldots) = R(\beta_1, \beta_2, \ldots)$

2) $\alpha = \beta,\ \beta = \gamma\ \rightarrow\ \alpha = \gamma$

3) $\rightarrow\ \alpha = \alpha$

4) $\alpha = \beta\ \rightarrow\ \beta = \alpha$

These are entirely analogous to the taxonomic inference rules except for the addition of rule (4).

The following theorem states that congruence closure can be reduced to taxonomic closure:

Theorem 4 *If a congruence sentence $\xi_=$ can be derived via the congruence closure inference rules from a set of congruence sentences $\Sigma_=$, then the taxonomic sentences $\xi_\subseteq$ and $\xi_\supseteq$ can be derived via the taxonomic closure inference rules from the taxonomic sentences $\Sigma_\subseteq \cup \Sigma_\supseteq$. Here $\zeta_\subseteq$ represents $\zeta_=$ with $=$ replaced by $\subseteq$ and $\zeta_\supseteq$ represents $\zeta_=$ with $=$ replaced by $\subseteq$ and the operands exchanged.*

Proof If $\zeta_=$ can be derived from $\vartheta^1_=, \vartheta^2_=, \ldots$ via congrence closure rules (1), (2), or (3), then both $\zeta_\subseteq$ and $\zeta_\supseteq$ can be derived from $\vartheta^1_\subseteq, \vartheta^2_\subseteq, \ldots$ and $\vartheta^1_\supseteq, \vartheta^2_\supseteq, \ldots$ via the analogous taxonomic closure inference rules. Furthermore, any inference via congruence closure rule (4) becomes a null inference when the congruence sentences are each mapped into a pair of taxonomic sentences. The theorem then follows by induction on the length of the derivation. ■

Note that this reduction can be carried out in linear time with a fixed amount of work space. Congruence closure can thus be solved in $O(n^3)$ time by applying the taxonomic closure algorithm. Downey, Sethi, and Tarjan [4] give a much better algorithm, however, which runs in $O(n \log n)$ time. Of more significance is that negative results for congruence closure apply to taxonomic closure as well.

P-Completeness of Monadic Congruence Closure

The congruence closure decision problem was shown to be log-space complete for P by Kozen[2] [6]. I will here extend this result to the *monadic* congruence closure decision problem, in which only function applications with a single argument are permitted. It follows that congruence closure, taxonomic closure, the taxonomic inference system of McAllester, *et al* [10], and the problem of inference with "Montague literals" of McAllester and Givan [9] are all P-complete, even if only monadic function application is permitted (as is always the case with Montague literals).

[2]It was seen by him as "the word problem for a finitely-presented algebra"

Theorem 5 *The monadic congruence closure decision problem is complete for P under log-space reductions.*

Proof I will show how to reduce the circuit value problem to monadic congruence closure. The circuit value problem was shown to be log-space complete for P by Ladner [7].

Input for the circuit value problem consists of the following:

- A list of input signals, numbered $1, 2, \ldots, i$, each with a Boolean value.
- A list of gates, numbered $i+1, i+2, \ldots, o$, giving the type of each gate (AND, OR, or NOT) and the numbers of the gate's input signals (which must precede it). The gate's output signal is represented by the gate number.

The problem is to determine whether the output of the last gate, o, is true or false.

An instance of the circuit value problem can be reduced to an instance of the monadic congruence closure decision problem using constants T and F to represent "true" and "false", a set of constants V_j to represent the values of the signals, a set of monadic functions AND_j whose values will be the AND of their argument with V_j, a set of monadic functions OR_j whose values will be the OR of their argument with V_j, and a monadic function NOT,

The reduction is performed as follows:

1) For each input signal, j, output either the sentence $V_j = T$ or the sentence $V_j = F$, expressing whether input j is true or false.

2) For each NOT gate, k, with input l, output the sentence $V_k = NOT(V_l)$.

3) For each AND gate, k, with inputs l and r, output the sentence $V_k = AND_l(V_r)$.

4) For each OR gate, k, with inputs l and r, output the sentence $V_k = OR_l(V_r)$.

5) For each input signal or gate of any type, with number k, output the sentences

$$AND_k(F) = F$$
$$AND_k(T) = V_k$$
$$OR_k(F) = V_k$$

$$OR_k(T) = T$$

Also output the sentences $NOT(T) = F$ and $NOT(F) = T$.

6) As the query for the monadic congruence closure problem, ask whether $V_o = T$ can be derived from the above sentences.

One can easily verify that the sentences output are consistent with the intended meanings of V_j, AND_j, etc. and that the value of V_o can always be derived from these premises. The query therefore will be satisfied if and only if the output, o, of the circuit is true. One can also easily see that the reduction requires at most logarithmic work space. ■

As is discussed later, a consequence of this result is that probably none of the inference systems discussed in this paper can be efficiently parallelized.

Completeness of Monadic Taxonomic Closure for 2NPDA

In this section I will show a relationship between monadic taxonomic closure and the problem of simulating pushdown automata, thereby shedding some light on the computational difficulty of taxonomic inference.

A *two-way nondeterministic pushdown automaton* (2NPDA) consists of a read-only input tape with a head that can move forward and backward, a pushdown stack holding symbols from some finite alphabet, and a control unit with a finite number of states. The operation of the 2NPDA is defined by a set of permitted transitions. Each transition applies when the control unit is in a particular state, the tape head is scanning a particular input symbol, and a particular symbol is on top of the stack. The transition specifies the new state of the control unit and whether to leave the tape head unmoved, move it one cell to the left, or move it one cell to the right. A *push* transition also specifies a symbol to be pushed onto the stack, while a *pop* transition specifies that the symbol on top of the stack is to be removed.

The 2NPDA is applied to an input string by placing the string on the input tape, with delimiter symbols at both ends, positioning the input tape head at the beginning, setting the stack to contain only the special symbol Z, and starting the control unit in state q_0. The 2NPDA *accepts* the input if there is some sequence of permitted transitions from this initial configuration that lead to the symbol Z being popped off the stack. Without loss of generality, I will assume that this can occur only when the input head is at the start of the tape and the control unit is in state q_f.

Theorem 6 *The problem of whether a particular 2NPDA accepts a given input of length n can be reduced to an instance of the monadic taxonomic closure decision problem of length $O(n \log n)$ using only $O(n \log n)$ time on a sequential RAM.*

Proof Operation of the 2NPDA on a given input will be simulated by a monadic taxonomic closure problem involving predicate symbols Q^i_j and relation symbols S_k. The intended interpretations of these symbols are as follows:

- $\hat{Q}^i_j$ The singleton set consisting of the pair (i, j) with i representing an input head position and j representing a state of the control unit.
- $\hat{S}_k$ The relation consisting of all tuples $< (i, j), (i', j') >$ such that if the 2NPDA were started with the input head at location i and the control unit in state j, with k on the top of the stack, it might, by following permitted transitions, pop the symbol k off the stack while leaving the input head at position i' and entering state j'.

The term $S_k(Q^i_j)$ will consequently be interpreted as the set of all pairs, (i', j'), such that if the 2NPDA were started on the given input with the head at position i, the control unit in state j, and the stack containing k, it might eventually pop k off the stack with the head at position i' and the control unit in state j'.

The reduction is performed as follows. For every possible position, i, of the input head ($0 \leq i \leq n+1$) and for every control unit state, j, and top-of-stack symbol, k, determine which transitions are permitted *for the given input string.* For every permitted push transition, output the sentence

$$S_k(S_{k'}(Q^{i'}_{j'})) \subseteq S_k(Q^i_j)$$

where i' is the new input head position (either $i-1$, i, or $i+1$), j' is the new state of the control unit, and k' is the symbol pushed onto the stack. For every permitted pop transition, output the sentence

$$Q^{i'}_{j'} \subseteq S_k(Q^i_j)$$

where i' and j' are as above. Finally, as the query, ask whether the sentence $Q^0_{q_f} \subseteq S_Z(Q^0_{q_0})$ can be derived from these premises.

For each input head position, this procedure outputs a bounded number of sentences. Each sentence contains occurrences of the symbols Q^i_j, whose number is proportional to n. These symbols can be encoded in $\log n$ space and time, leading to $O(n \log n)$ bounds for the total size of the output and for the total time required.

It is left as an excercise for the reader to verify that the sentences are valid for the intended interpretation and that they are sufficient to allow all execution paths of the 2NPDA to be inferred. ■

The above theorem shows, for example, that if the monadic taxonomic closure decision problem, or any of the more general taxonomic inference problems, could be solved in $O(n^2)$ time on a RAM, then any language recognizable by a 2NPDA could be recognized in $O(n^2 \log^2 n)$ time on a RAM.

A converse theorem also holds:

Theorem 7 *The monadic taxonomic closure decision problem can be solved by a 2NPDA.*

Proof Any proof of a sentence $\alpha \subseteq \beta$ from premises Σ using the taxonomic inference rules (1), (2), and (3) can be converted to a chain of the form

$$\alpha = \kappa_1 \subseteq \kappa_2 \subseteq \ldots \subseteq \kappa_{n-1} \subseteq \kappa_n = \beta$$

where each κ_i is obtained from κ_{i-1} by substituting the term δ for some occurrence of the sub-term γ, with the sentence $\gamma \subseteq \delta$ being in Σ. These substitutions can be justified by repeated application of inference rule (1). In the case of monadic taxonomic sentences, substitutions will be of the form

$$R_1 \ldots R_i R_{i+1} \ldots R_n P \rightarrow R_1 \ldots R_i R'_{i+1} \ldots R'_m P'$$

where the sentence $R_{i+1} \ldots R_n P \subseteq R'_{i+1} \ldots R'_m P'$ is in Σ. Here, the R_j and R'_j are relation symbols, while P and P' are predicate symbols. I have omitted parentheses from the monadic applications, as I assume will also be the case with the input for the 2NPDA.

Using these observations, the following procedure can determine whether the query $\alpha \subseteq \beta$ follows from premises Σ:

1) Locate α in the input and push its symbols onto the stack in order. This leaves the predicate at the end of the term on top of the stack.

2) Scan all the premises looking for any of the form $\gamma \subseteq \delta$ with γ matching the top portion of the stack. If there is more than one such premise, nondeterministically choose one. If there is no such premise, then do not continue the computation.

3) Replace the portion of the stack matching γ with δ.

4) Locate β in the input and determine whether it is the same as the contents

of the stack. If so, accept the input. Otherwise, go back to step (2).

It is clear that one of the computations nondeterministically specified by the above procedure will accept the input if and only if the query follows from the premises. Furthermore, the procedure can be implemented on a 2NPDA. The only tricky parts are the checks for whether the stack contents match a sub-string of the input. These can be implemented by popping off stack elements as the sub-string is scanned, and restoring them from the sub-string itself when a mis-match is found or the end of the sub-string is reached. ■

Since the best known algorithm for simulating a 2NPDA requires $O(n^3)$ time [1], the above result does not, unfortunately, provide any improvement on theorem (3).

The above two theorems can be strengthened, though the result is perhaps mostly of interest from the viewpoint of automata theory. The stronger result is that a particular encoding of the monadic taxonomic closure decision problem is complete for the class of languages recognizable by a 2NPDA.

The appropriate reductions for this class are *homomorphic mappings*, in which each symbol of the input is mapped to a sequence of output symbols independently of the context in which the symbol occurs. To permit such a context-independent mapping, it will be necessary to use a *relative* encoding of the taxonomic closure predicate symbols. The obvious method of encoding a potentially unbounded number of symbols (assumed above) is via some absolute numbering scheme. Input for an instance of the taxonomic closure problem might look like the following, for instance, with all but the predicate symbols replaced with ellipses:

$$\ldots\ P^2 \ldots\ Q^5 \ldots\ P^4 \ldots\ Q^4 \ldots\ Q^5 \ldots\ P^2 \ldots$$

Here, P and Q are two symbol bases from which symbols are built by appending indexes.

With relative encoding, an symbol's index is specified by giving its offset from the index of the preceding symbol, expressed via a series of "+" or "−" symbols. The index zero is taken as the starting point at the beginning of the input. The above example would be relatively encoded as follows:

$$\ldots\ P^{++} \ldots\ Q^{+++} \ldots\ P^{-} \ldots\ Q \ldots\ Q^{+} \ldots\ P^{---} \ldots$$

Theorem 8 *The monadic taxonomic closure decision problem with relative encoding of predicate symbols is complete under homomorphic reductions for the* *class of languages recognizable by a 2NPDA.*

Proof The reduction used for theorem (6) will be adapted, with the symbols Q_j^i being relatively encoded with respect to i, the Q_j being taken as symbol bases.

The premises associated with head position i will be output as a group, in order of increasing i. The preceding output will have been such that the last predicate symbol preceding the group will have had index i, allowing the premises to be independent of preceding context. Following each group of premises, some number of innocuous reflexive sentences of the form $Q_j^+ \subseteq Q_j$ will be output to ensure that the predicate symbol index is set to $i+1$ for the next group. The query will be output at the beginning, as part of the group for head position 0.

With this scheme, the set of sentences output for group i does not, in fact, depend on i itself, but only on the input symbol located at position i — i.e. the mapping is homomorphic.

It is also necessary to show that the problem can be solved by a 2NPDA when relative encoding is used. The algorithm of theorem (7) can still be applied, with the modification that the term held on the stack must be represented in the form in which it would appear at the point in the input where the read head is currently located. Whenever the 2NPDA scans forward over a "+" or backward over a "−" (other than temporarily as part of a comparison operation), a "+" is removed from the top of stack if one is present, or a "−" is pushed on if no "+" is present. The reverse is done when scanning forward over a "−" or backward over a "+". ■

Since homomorphic mappings can be embedded in the finite control of an automaton[3], this theorem allows one to conclude, for example, that if the monadic taxonomic closure decision problem with relative encoding could be solved by a two-way deterministic pushdown automaton (2DPDA), then all languages recognized by a 2NPDA could be recognized by a 2DPDA as well.

Finally, the following result can be obtained:

Theorem 9 *The problem of whether a particular 2DPDA accepts a given input of length n can be reduced to an instance of the monadic congruence closure decision problem of length $O(n \log n)$ using only $O(n \log n)$ time on a sequential RAM.*

Proof A reduction entirely analogous to that for theorem (6) is performed, except that the sentences output assert set equality rather than set containment.

[3]See [5] p. 61, where this is done for finite automata.

This is valid, since a deterministic pushdown automaton can have at most one permitted transition for a given combination of state, head position, and stack symbol. ■

A linear-time algorithm for simulating a 2DPDA was found by Cook [3], so the above theorem does not improve on known results. It would be interesting to know whether an analogue of theorem (7) holds — i.e. whether monadic congruence closure can be solved by a 2DPDA.

Discussion

The class NC of problems solvable in polylogarithmic ($O(\log^k n)$) time using a polynomial number of processors is often taken as defining the set of efficiently parallelizable computations [12]. It is known that problems that are log-space complete for P are not in NC unless all problems in P are in NC [2], which is thought to be unlikely. The P-completeness of monadic congruence closure thus makes it unlikely that any of the inference systems discussed in this paper have efficient parallel decision algorithms.

From an engineering viewpoint, this result poses problems for any attempt to use taxonomic inference in a system of "common sense" reasoning, where the size of the knowledge base would be enormous. Even in the more restricted domain of mathematical theorem proving [8], the "lemma library" could become very large.

There may, of course, be many applications in which the current $O(n^3)$ sequential algorithm is adequate, and more that would be feasible if an improved sequential algorithm were found. Theorem (6) shows that significant progress here would also produce a better algorithm for simulating a 2NPDA. The reader is free to take this either as an additional motive to work on the problem, or as evidence that there may be no solution, since the $O(n^3)$ 2NPDA simulation of Aho, Hopcroft, and Ullmann [1] has apparently not been improved on since 1968. In this connection, one should note that all context free languages can be recognized by a 2NPDA. Context-free language recognition has been intensively studied, with the best general algorithm found being that of Valiant [13], which requires the same time as matrix multiplication.

It could be argued that the worst-case approach to the analysis of these inference systems is overly pessimistic for many applications. One might also consider the real problem to be maintenance of an incrementally-built knowledge base in a form permitting fast response to queries. A straightforward adaptation of the algorithm of theorem (3), for example, allows queries of size q to be answered in

$O(p^2q + pq^2 + q^3)$ time, where p is the size of the premises, after a pre-processing stage requiring $O(p^3)$ time.

McAllester and Givan [9] show how the $O(n^3)$ taxonomic inference procedure of [10] can be extended to encompass sentences built from "Montague literals" that correspond to certain English sentences typified by "every man loves some woman". They argue that the existence of this inference procedure may provide a functional explanation for this syntactic feature of English.

Sentences built from Montague literals are more expressive than monadic congruence closure, however, and hence the inference procedure for them is P-complete. The fact that neurons interact at time scales of a few milliseconds and apparently compute relatively simple functions internally leads one to expect a cognitively plausible inference system to be parallelizable. In [8], McAllester notes that congruence closure is indeed capable of "superhuman" inferences, and proposes a weaker scheme for inference about equality that is both more parallelizable and more of a match for human performance. An inference procedure for Montague literals that shared these characteristics would be of interest.

Acknowledgement

I thank Charles Elkan for introducing me to the work of McAllester, *et al* and for many helpful discussions.

References

[1] Aho, A. V., Hopcroft, J. E, and Ullmann, J. D. (1968) Time and tape complexity of pushdown automaton languages, *Information and Control*, vol. 13 no. 3, pp. 186-206.

[2] Borodin, A. (1977) On relating time and space to size and depth, *SIAM Journal of Computing*, vol. 6 no. 4 (December 1977), pp. 733-744.

[3] Cook, S. A. (1971) Linear time simulation of deterministic two-way pushdown automata, *Proceedings of the 1971 IFIP Congress*, pp. 75-80.

[4] Downey, P. J., Sethi, R. and Tarjan, R. E. (1980) Variations on the common sub-expression problem, *Journal of the ACM*, vol. 27 no. 4 (October 1980), pp. 758-771.

[5] Hopcroft, J. E. and Ullman, J. D. (1979) *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley.

[6] Kozen, D. (1977) Complexity of finitely presented algebras, *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, Boulder, Colorado (May 1977), pp. 164-177.

[7] Ladner, R. E. (1975) The circuit value problem is log space complete for P, *SIGACT News*, vol. 7 no. 1 (January 1975), pp. 18-20.

[8] McAllester, D. (1989) *ONTIC: A Knowledge Representation System for Mathematics.*

[9] McAllester, D. and Givan, B. (1989) Natural language syntax and first order inference, Massachusetts Institute of Technology AI memo no. 1176.

[10] McAllester, D., Givan, B. and Fatima, T. (1989) Taxonomic syntax for first order inference, *Proceeding of the First International Conference on Principles of Knowledge Representation and Reasoning*, pp. 289-300.

[11] Nelson, G. and Oppen, D. C. (1980) Fast decision procedures based on congruence closure. *Journal of the ACM*, vol. 27 no. 2 (April 1980), pp. 356-364.

[12] Pippenger, N. ?

[13] Valiant, L. (1975) General context-free recognition in less than cubic time, *Journal of Computer and System Science*, vol. 10 no. 2 (April 1975), pp. 308-315.

Appendix — Proof of Soundness and Completeness

The following theorem, asserting that the taxonomic inference system is semantically sound and complete, was stated earlier:

Theorem 1 *For any set of sentences Σ and any sentence ξ, $\Sigma \models \xi$ if and only if $\Sigma \vdash \xi$.*

Proof For each of the inference rules, one can verify that if the premises of the rule are true under some interpretation, then the conclusion is also true. This is easily seen for rules (2) and (3). Looking at rule (1), for any $x \in I(R(\alpha_1, \alpha_2, \ldots))$ there must exist $u_1, u_2, \ldots$ for which $\hat{R}(u_1, u_2, \ldots, x)$ and $u_1 \in I(\alpha_1)$, $u_2 \in I(\alpha_2)$, etc. If the premises of rule (1) are true, it will also be that $u_1 \in I(\beta_1)$, $u_2 \in I(\beta_2)$, etc. and hence $x \in I(R(\beta_1, \beta_2, \ldots)$, showing that the conclusion of the rule is true as well.

If $\Sigma \vdash \xi$, it follows by induction on the length of the proof that if the sentences of Σ are true under an interpretation, then ξ is also true under that interpretation

— i.e. that $\Sigma \models \xi$. Thus the inference rules are sound.

To prove completeness, an interpretation, M_Σ, will be constructed under which all sentences in Σ are true. By the definition of entailment, if $\Sigma \models \xi$, then ξ must also true under M_Σ, and the construction of M_Σ will be such that this in turn implies $\Sigma \vdash \xi$.

The interpretation M_Σ is defined as follows:

- Let the domain of M_Σ, D, be the set of terms built from the predicate symbols, P_i, and relation symbols, R_i.
- For each predicate symbol, P_i, let

  $$\hat{P}_i = \{\, \lambda \in D \mid \Sigma \vdash \lambda \subseteq P_i \,\}$$

- For each relation symbol, R_i, of arity n, let

  $$\hat{R}_i = \{\, < \kappa_1, \ldots, \kappa_{n-1}, \lambda > \in D^n \mid \Sigma \vdash \lambda \subseteq R_i(\kappa_1, \ldots, \kappa_{n-1}) \,\}$$

Under this interpretation, $I(\alpha) = \{\, \lambda \mid \Sigma \vdash \lambda \subseteq \alpha \,\}$, for any term α. This can be seen by induction. The statement obviously holds for terms that are simply predicate symbols. Furthermore, if the statement holds for terms $\alpha_1, \ldots, \alpha_{n-1}$ then for any λ:

$$\begin{aligned}
\lambda \in I(R(\alpha_1, \alpha_2, \ldots)) &\Rightarrow (\exists \beta_1, \beta_2, \ldots)\ \beta_1 \in I(\alpha_1)\ \&\ \beta_2 \in I(\alpha_2)\ \&\ \cdots \\
&\qquad \&\ \hat{R}(\beta_1, \beta_2, \ldots, \lambda) \\
&\Rightarrow (\exists \beta_1, \beta_2, \ldots)\ \Sigma \vdash \beta_1 \subseteq \alpha_1\ \&\ \Sigma \vdash \beta_2 \subseteq \alpha_2\ \&\ \cdots \\
&\qquad \&\ \Sigma \vdash \lambda \subseteq R(\beta_1, \beta_2, \ldots) \\
&\Rightarrow \Sigma \vdash R(\beta_1, \beta_2, \ldots) \subseteq R(\alpha_1, \alpha_1, \ldots) \\
&\qquad \&\ \Sigma \vdash \lambda \subseteq R(\beta_1, \beta_2, \ldots) \\
&\Rightarrow \Sigma \vdash \lambda \subseteq R(\alpha_1, \alpha_2, \ldots)
\end{aligned}$$

where the last two implications apply inference rules (1) and (2), respectively. In the other direction:

$$\begin{aligned}
\Sigma \vdash \lambda \subseteq R(\alpha_1, \alpha_2, \ldots) &\Rightarrow \Sigma \vdash \alpha_1 \subseteq \alpha_1\ \&\ \Sigma \vdash \alpha_2 \subseteq \alpha_2\ \&\ \cdots \\
&\qquad \&\ \Sigma \vdash \lambda \subseteq R(\alpha_1, \alpha_2, \ldots) \\
&\Rightarrow \alpha_1 \in I(\alpha_1)\ \&\ \alpha_2 \in I(\alpha_2)\ \&\ \cdots \\
&\qquad \&\ \hat{R}(\alpha_1, \alpha_2, \ldots, \lambda) \\
&\Rightarrow \lambda \in I(R(\alpha_1, \alpha_2, \ldots))
\end{aligned}$$

where here use has been made of inference rule (3). Hence, for all α, $I(\alpha) = \{\, \lambda \mid \Sigma \vdash \lambda \subseteq \alpha \,\}$.

Any sentence $\alpha \subseteq \beta$ that is in Σ is now seen to be true under M_Σ, since $\lambda \in I(\alpha)$ implies that $\Sigma \vdash \lambda \subseteq \alpha$, which due to inference rule (2) implies that $\Sigma \vdash \lambda \subseteq \beta$, and finally $\lambda \in I(\beta)$. Furthermore any sentence $\alpha \subseteq \beta$ that is true under M_Σ can be derived from Σ, since by rule (3) $\Sigma \vdash \alpha \subseteq \alpha$, hence $\alpha \in I(\alpha)$, then $\alpha \in I(\beta)$ by the definition of truth, and finally $\Sigma \vdash \alpha \subseteq \beta$.

These two statements imply the desired result: If $\Sigma \models \xi$, then ξ is true under all interpretations for which the sentences of Σ are true, hence ξ is true under M_Σ, and therefore $\Sigma \vdash \xi$. ■