Progressing the State-of-the-art in Grammatical Inference by Competition

The Omphalos Context-Free Language Learning Competition

Bradford Starkie [a,*], François Coste [b] and Menno van Zaanen [c]

[a] *Telstra Research Laboratories*
770 Blackburn Rd Clayton
Melbourne, Victoria
Australia 3127
E-mail: Brad.Starkie@telstra.com.au
[b] *IRISA*
Campus de Beaulieu
35042 Rennes
France
E-mail: francois.coste@irisa.fr
[c] *Tilburg University*
Postbus 90153
5000LE, Tilburg
the Netherlands
E-mail: mvzaanen@uvt.nl

This paper describes the Omphalos Context-Free Language Learning Competition held as part of the International Colloquium on Grammatical Inference 2004. After the success of the Abbadingo Competition on the better known task of learning regular languages, the competition was created in an effort to promote the development of new and better grammatical inference algorithms for context-free languages, to provide a forum for the comparison of different grammatical inference algorithms and to gain insight into the current state-of-the-art of context-free grammatical inference algorithms.

This paper discusses design issues and decisions made when creating the competition, leading to the introduction of a new complexity measure developed to estimate the difficulty of learning a context-free grammar. It presents also the results of the competition and lessons learned.

Keywords: Grammatical Inference, Context-Free Grammars, Competition

*Corresponding author.

AI Communications
ISSN 0921-7126, IOS Press.

1. Introduction

Research performed in the field of Grammatical Inference (GI) is, as the name suggests, directed towards learning grammars from "sequential" data (i.e. sequences, words or strings but also more complicated structures such as trees, terms or even graphs). The induced, or inferred, grammars can then be used to classify unseen data, compress this data or provide some suitable model for this data. The Omphalos competition focussed upon the inference of Context-Free Grammars (CFG's).

Grammatical inference is a well established research field in Artificial Intelligence as it dates back to the 1960s. Gold [1] originated this study motivated by the construction of formal model of human language acquisition. Since this seminal paper, grammatical inference has been investigated, more or less independently, within many research fields, including machine learning, computational learning theory, structural and syntactic pattern recognition, computational linguistics and computational biology. A good survey of the field is presented in [2] while [3] proposes a recent guide into its bibliography.

Formal grammatical inference is predominantly concerned with the learnability of specific types of grammars. Learnability in this type of work can be defined in different ways. For example, the well-known work by [1] defines learnability in a game theoretic way. The idea is that *in the limit* a learner has to find the same grammar used by the teacher to generate examples of the language described by the grammar. The setting of active learning, with the possibility of asking queries to the teacher has also been studied [4]. A different definition of learnability is called probably approximately correct (PAC) learning [5], where the goal is to minimize the probability of learning an incorrect grammar. Within this field, proofs show that

certain classes of grammars can be learnt according to the set of learnability criteria.

When considering efficiency, results are mainly negative even for the simplest class of the Chomsky hierarchy if the presentation of the examples is arbitrary. Gold [6] proved that the problem of finding the smallest deterministic finite automata (DFA) consistent with a given finite sample of positive and negative examples is NP-hard. Moreover, Pitt and Warmuth [7] showed that no polynomial algorithm could be guaranteed to find an approximation (in the size) of this target automata. In the PAC setting, they also showed that the problem of polynomially approximate predictability of the class of DFA is hard [8], indeed this problem can be linked with solving cryptographic problems believed to be intractable [9]. To allow efficient learnability, one has to assume that "good" examples are provided to the learner: simple distributions in the PACS setting [10] or presence of samples containing *characteristic sets* formalized by de la Higuera [11] in his model of *identification in the limit from polynomial time and data*. This later model enables all languages that can be described using a DFA to be learnt but only when the training set contains a characteristic set of sentences. Note that even in this setting, not all context-free languages can be identified.

Due to the hardness of the task, the main algorithms developed for the grammatical inference of context-free languages are either algorithms devoted to specific subclasses of grammars [12,13,14,15,16] or heuristic approaches [17,18,19] which are both difficult to compare from a theoretical or computational point of view. One of the more objective ways of comparing these various approaches is to test their generalization performance on given data. Several benchmarks have been constructed by the authors to test their own algorithms[1] but to really compare these various approaches, "neutral" benchmarks are needed. This has been one of the achievements of grammatical inference competitions.

Here, we will describe *Omphalos*, a context-free language learning competition held in conjunction with the International Colloquium on Grammatical Inference 2004.[2] Omphalos is the latest competition in a line of competitions within the GI field. Whereas Abbadingo[3] and its follow-on Gowachin[4], or the GECCO contest from noisy data[5] were directed towards learning deterministic finite automata, which can describe the class of regular languages, Omphalos is the first GI competition aimed at learning context-free languages.

Motivated by the progress permitted by the Abbadingo competition [20], the competition was set up with the following aims:

- to promote the development of new and better grammatical inference (GI) algorithms just beyond the current state-of-the-art in grammatical inference,
- to provide a forum in which the performance of different grammatical inference algorithms can be compared on a given task, and
- to provide an indicative measure of the complexity of grammatical inference problems that can be solved with the current state-of-the-art techniques.

The rest of the article is divided as follows. We will start with a description of context-free grammars for those unfamiliar with the concept. In that section we will also define the notation used in this paper. We will then describe the competition task of Omphalos, in a section where we treat design issues, the actual setup of the competition and different evaluation methods.

We propose in the next section a new complexity measure to estimate the difficulty of learning a context-free grammar. The article will end with the results of the competition, including the contribution of the competition to advancing the state-of-the-art and a discussion on the lessons that have been learnt from the competition.

1.1. Context-Free Grammars

A context-free grammar can be represented by a 4-tuple $G = (N, \Sigma, \Pi, \mathrm{S})$ where N is the alphabet of non-terminal symbols; Σ is the alphabet of terminal symbols such that $N \cap \Sigma = \emptyset$; Π is a finite set of productions P of the form "$\mathrm{L} \rightarrow \mathrm{s}_1\mathrm{s}_2 \ldots \mathrm{s}_n$" where $\mathrm{L} \in N$ and each $\mathrm{s}_i \in (N \cup \Sigma)$; and S is a special non-terminal called the start symbol that represents all of the sentences generated by G. For the

[1] See for instance this GI benchmark repository http://www.irisa.fr/symbiose/people/coste/gi_benchs.html.

[2] The competition data will continue to be available after the completion of the competition and can be accessed via http://www.irisa.fr/Omphalos/.

[3] See http://abbadingo.cs.unm.edu/.

[4] See http://www.irisa.fr/Gowachin/.

[5] See http://cswww.essex.ac.uk/staff/sml/gecco/NoisyDFA.html.

sake of brevity, a context-free grammar is sometimes described purely in terms of its production rules. This is because as long as the start symbol is known the values of N and Σ can be extracted from the production rules. All example grammars in this paper will use the start symbol S.

For example consider the following grammar.

Grammar 1.

S → i'd like to fly X
X → to LOC
X → from LOC to LOC
LOC → melbourne
LOC → sydney

Valid sentences of a language described by a grammar can be created by selecting one of the rules that has the start symbol on its left-hand side. Each non-terminal N_i on the right-hand side of a rule is then expanded by replacing N_i with the right-hand side of a rule with N_i on its left-hand side. The process is repeated until no further non-terminals can be expanded.

The notation "A $\Rightarrow^*$ **w**" denotes that A can be expanded via zero or more applications of a production rule to become **w**.

For instance according to the Grammar 1:

S $\Rightarrow^*$ i'd like to fly from melbourne to sydney

The sequence of rules used to generate a sentence can be represented by a structure known as a derivation tree, that can be represented by either by a tree diagram (Figure 1) or series of bracketings (Figure 2).

In this document we will use the following notations.

1. Lowercase letters represent terminal symbols, e.g., "a", "b".
2. Uppercase symbols represent non-terminal symbols, e.g., "A", "B". The symbol S will be reserved to represent the start symbol.
3. Bold lowercase letters represent sequences of one of more terminal symbols, e.g., "**a**", "**b**".
4. Italicized lowercase letters represent sequences of zero of more terminal symbols, e.g., "a", "b".
5. The symbol ε is used to represent the empty string.
6. Bold uppercase letters represent sequences of one or more terminal or non-terminal symbols, e.g., "**A**", "**B**".
7. The notation L(G) will be used to denote the language defined by the grammar G.

A special subclass of context-free grammar is the right linear grammar. In a right linear grammar, all production rules are either of the form "A → bC", "D → e", or "F → ε".

Any language that can be described using a context-free grammar is known as a context-free language. Similarly any language that can be described using a right linear grammar is known as a regular language. The class of all regular languages is a proper subset of the class of all context-free languages.

2. Design issues

Apart from the class of grammars to be learnt, several design issues have to be decided upon. The main issues were:

Format of training data For the participants of the competition to learn an unknown language, example sentences from that language need to be made available to them. There are many design choices to made about the format of these example sentences, for instance, the amount of data, the type of data, completeness, amount of noise in the data, etc.

Method of evaluation To be able to determine a winner of the competition, a means of evaluating the results needed to be decided upon. There are several ways of doing this, with certain problems associated with each possibility. We could, for instance, judge entries by measuring the difference between the inferred language and the target language, or, alternatively, by measuring the difference between derivation trees assigned to certain sentences by the inferred grammar and the corresponding derivation trees of the target grammar.

Complexity of tasks One of the goals of the competition was to determine and advance the state-of-the-art in grammatical inference. Therefore the competition tasks should be selected so as to be neither too simple, nor too difficult to solve. To be able to rank the competition problems, the complexity of the learning task should be quantifiable. This design issue is, of course, closely related to several aspects of the held-out data.

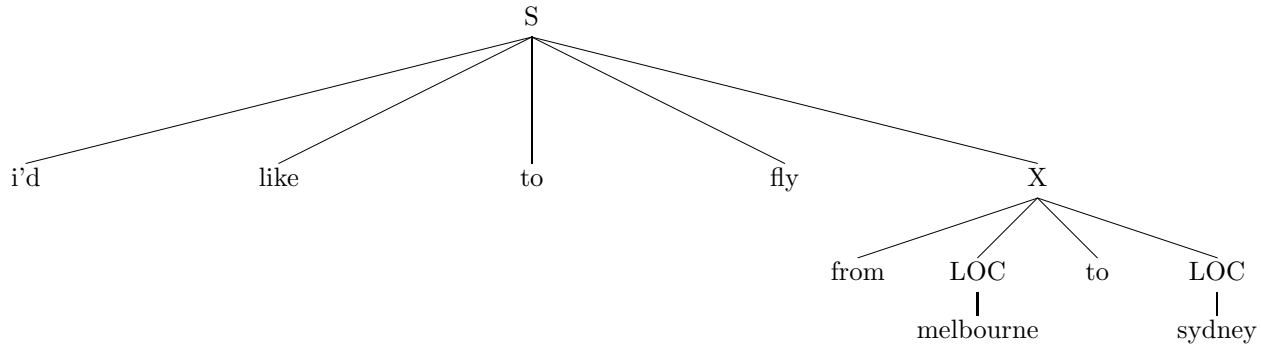

Fig. 1. Example parse tree using the grammar of Grammar 1.

(S i'd like to fly (X from (LOC melbourne) to (LOC sydney)))

Fig. 2. Alternative parse tree notation.

2.1. Format of Training Data

On a high level, one needs to decide the type of information that is given to the learning algorithm. There are several options, among others:

positive examples only a set of sentences that are part of the language,

positive and negative examples a set of sentences where each is labelled to be a member of the language or not,

(partially) structured examples a set of tree structures denoting the (partial) analysis of the sentence according to the underlying grammar.

It is known that (within the framework of "identification in the limit"), many interesting classes of grammars are not learnable from positive data only. Context-free grammars are one of them [1]. However, [21] gives an enumerative algorithm that identifies Stochastic Context Free Grammars (SCFGs) in the limit with probability one from stochastic data, i.e. data generated by an SCFG. The type and the amount of queries needed to learn context-free grammars is still an open problem [4].

The goal of the competition is not to see whether context-free grammars are learnable or not, but to investigate the interaction between the learning algorithms and the complexity of the grammars, the amount and the type of data available. Many parameters define the possibility of whether or not context-free grammars are learnable.

In this competition, it was decided to provide both positive or positive and negative examples. Even though learning a grammar from structured examples is not trivial, (partially) structured examples are too closely related to an underlying grammar. Using positive and negative examples is more in line with the previous (regular language) competitions, and it is also possible for systems to ignore the negative examples, if they want to claim to learn the grammar using positive examples only.

Apart from the type of training data, there are still a number of aspects of the data that are important for the setting of the problems. These aspects will be described below in section 3, where the complexity of specific tasks is discussed.

2.2. Method of Evaluation

In order to find out when a system is successful in learning a grammar, a means of evaluation needs to be decided upon. We must have a way of knowing whether the grammar that underlies the training sentences has been learnt. There are several ways of tackling this problem. The actual method should be automatic, objective, and easy to use.

Before we can decide upon an evaluation method, we must first decide what exactly we want to be able to learn. A specific grammar defines a fixed language that possibly contains an infinite number of sentences. The sentences can be generated by combining the grammar rules, which can be seen

as a tree structure on top of the language. The *string language* is now the set of plain sentences that are part of the language and the *tree language* is the set of sentences together with their derivation trees. There are often different ways to generate the same sentence using a grammar, which means that one sentence can have different trees. This is an important aspect in evaluation, because we need to know exactly what the goal, or the output of the GI system should be.

There are many possible approaches to the evaluation of grammatical inference systems. Here we will give an overview of several ways to evaluate GI systems together with a discussion on the advantages and disadvantages of each [22].

2.2.1. Rebuilding known grammars

Starting from a pre-defined grammar, a set of training sentences is generated. Using this set of sentences, the GI system should recreate the original grammar. The output grammar of the GI system is then compared against the original grammar.

There are several advantages of this approach.

- The grammar can be created by hand, having specific properties that might be difficult to learn by grammatical inference systems.
- The complexity of the grammar can be varied if needed by choosing a different grammar.
- It is easy to vary the amount of training data, because if necessary, more data can be generated.
- The generation process can be adjusted, allowing, for example an ordering in the training data or a preference for sentences with certain aspects.

However, there are also problems with this approach.

- For most languages, there exists more than one grammar that generates the same string language. Although all regular grammars can be transformed into a canonical form, no such canonical form exists for context-free grammars. This implies that it is impossible to compare context-free grammars directly and the only way to know for sure if two grammars are the same is to do manual investigation or to compare the set of sentences that can be generated by the grammars. Unfortunately, in the general case a context-free grammar can have an infinite number of sentences and comparing two grammars may not be possible. It is definitely the case that this cannot be performed automatically using the current state-of-the-art.
- The flexibility of sentence generation can also be seen as a disadvantage. A decision has to be made on how exactly the sentences are generated. Not only may this influence the applicability of certain GI systems, it also introduces a new design choice: should information on the generation method be available to the GI systems?

2.2.2. Comparison of treebanks

This evaluation approach is based on a treebank, an annotated set of trees. A tree in this context is a sentence in the language combined with its derivation according to the grammar. The GI system may use the plain sentences from the treebank to try and structure these sentences. The inferred trees (which indirectly describes the grammar) are compared against the structure of the original treebank. A distance measure is then used to rank the similarity between the derivation trees assigned to sentences by the GI system and derivation trees that are in the original treebank.

Compared to the previous approach, the comparison has certain advantages.

- An automated comparison of the output of the GI system against the original treebank is possible. Comparing tree structures can be done automatically.
- Even partial grammatical equivalence can be measured by the distance metric. Once again, because for most languages there is more than one grammar describing that language, a grammar submitted by a competitor may describe the target language exactly, but would rank poorly according to the distance measure.

Again, there are certain problems with this approach.

- When ambiguity occurs in either the target or inferred grammar, multiple solutions are then possible. The GI system must now disambiguate between the multiple structurings (even though the string language of both grammars is the same).

– Related to the previous point, in addition to learning a grammar, this approach also needs a way to create structured versions of the input sentences.

2.2.3. Classification of unseen examples

The previous evaluation methods concentrate on measuring the structure that is learnt by the GI system. This can be a grammar as in the first method or the structure of sentences as in the second method. Because these methods both have problems related to comparing the structure, we also describe two methods that do not necessarily depend on the actual structure of the grammar or the sentences.

In the classification of unseen examples approach, the GI systems receive unstructured training data. This training data can be labelled as positive (i.e. part of the language) only or positive and negative (i.e. not part of the language). The sentences in the language can be generated from a grammar describing the language, negative sentences need to be generated in another way. Next, the GI system should assign language membership information to unseen (unlabelled) test data. In other words, the system must say for each test sentence whether or not it is contained in the language described by the underlying grammar.

There are several advantages of this approach.

– The evaluation is unaffected by the actual structure of the grammar. This removes the problems with comparing grammars. Additionally, a wider variety of inference techniques can be used by competitors which enables the competition to compare the results of a wider range of techniques.
– The actual evaluation process is clear. The labels can be correct or incorrect and not partially correct (which may occur when tree structures are compared).

Of course, this approach has some problems as well.

– Since this task is essentially a classification task, no real grammatical information has to be learnt as such. In fact, generic machine learning classifiers can be used.
– If too little test data is used, it is hard to get a good idea on the actual performance of the GI systems.
– The way in which negative data is constructed for the purposes of testing has a major impact upon the ability of the test to measure the distance between the inferred grammar, and the target grammar. For instance, if the negative data varies greatly from the positive data, then it may be possible to correctly classify negative and positive data with very little actual knowledge of the target language. In contrast if the positive and negative examples are very similar then a much more accurate model of the target language is required to accurately classify the test sentences.

2.2.4. Precision and recall

Each GI system receives unstructured training data generated according to a target grammar. From this training data a grammar is inferred. Next, previously unseen sentences that are in the target language are provided to the system. The *percentage* of these sentences that are in the inferred grammar is then measured. This measure is known as the recall. Next sentences are generated according to the inferred grammar. The number of these sentences that exist in the target language is then measured. This measure is known as the precision. Recall and precision can then be merged into a single measure known as the F-score. The F-score is a measure of the similarity between the target and inferred languages.

Advantages of this approach are as follows.

– This approach is quite similar to the classification of unseen examples. The main difference is that it is more flexible; systems that do not perform a perfect classification can still be measured against each other.
– The precision and recall measures give a nice indication on the correctness and completeness of the systems. The F-score combines these measures together in one number, which allows for an easy and direct comparison of systems.

Of course, this approach has disadvantages as well.

– Once a test sentence has been used to measure recall it cannot be used to measure recall a second time. This because that test sentence can also be used as an additional training example.
– This technique requires that all grammatical inference systems be capable of generating example sentences.

2.2.5. Design choice

All these evaluation methods have problems when applied to a generic GI competition as described above. It was decided, however, that for the Omphalos competition, the technique of "classification of unseen examples" was to be used to identify the winner of the competition. There are several reasons for this. Firstly, this is the same evaluation method that is used for the Abbadingo [20] DFA (regular language) learning competition; therefore this method already has some acceptance in the GI community. Secondly, it allows for many GI systems to participate (no structure needs to be generated) and, lastly, automatic evaluation is possible.

For Omphalos, a competitor was considered to have solved a problem when the test sentences were classified correctly with 100% accuracy, compared to 99% accuracy for the Abbadingo competition. This stricter requirement was used in an effort to encourage the development of new truly context-free learning algorithms.

The main benefit of this technique is that it places few restrictions upon the techniques used to classify the data. In addition, if the inferred grammar describes the target language exactly it will classify the test examples exactly. The main disadvantage of this technique is that it is possible for a (non-GI) classifier to classify all of the test examples exactly without the classifier having an accurate model of the target language.

One way to overcome the problem that classifiers can be used, is to only consider the problem to be solved when the precision of the inferred grammar is greater than a threshold value. We have decided not to implement this additional constraint, since it is believed that if the test sets contain negative examples that are sufficiently close to positive examples, then classification accuracy is a suitable measure of how close the inferred grammar is to the target grammar.

3. Measuring the complexity of a learning task

Ideally, when constructing the learning tasks in the Omphalos competition it should be known that it is possible for a competitor to solve the problems. In addition it is desirable to know how difficult it would be for a competitor to solve those problems. To do this a measure of the complexity of a grammatical inference task was needed. Such a measure can be used to rank grammatical inference tasks not only within the competition, but can also be used to rank the complexity of GI problems that can be solved using state-of-the-art techniques at any point in time.

To determine the complexity of a grammatical inference task, a measure was derived by modelling the learning task as a brute force search. To do this a hypothetical grammatical inference algorithm called the *BruteForceLearner* algorithm was created. This algorithm can identify any context-free grammar in the limit in exponential time, using positive and negative data. The choice of modelling a grammatical inference algorithm that used both positive and negative data was taken because it is known that context-free languages cannot be identified without negative examples, using the identification in the limit learning model. The size of the hypothesis space considered by the *BruteForceLearner* algorithm was used as a measure of the complexity of the learning task. The model was also used to determine if the training data was sufficient to identify the target grammar, that is whether or not it was possible to solve the competition problems. The model also gave insight into the way in which the training data should be constructed. As far as it is known this work is the first to have quantified the size of a minimum hypothesis space for grammatical inference of all context-free languages, as well as being the first test that can be applied to determine if a set of training examples is sufficient for at least one algorithm to infer the target language.

The definition of the *BruteForceLearner* is given in Algorithm 3. This function makes use of the *PruneGrammars* and *ConstructAllGrammars* functions given in Algorithm 1 and 2 respectively.

Definition 1. *A context-free grammar is in Chomsky Normal Form (CNF) if all rules are of the form "$S \rightarrow \varepsilon$", "$A \rightarrow BC$" or "$A \rightarrow b$" where S is the start symbol, A, B, C are non-terminals, ε is the empty string and b is a terminal symbol.*

It will now be shown that the *BruteForcerLearner* algorithm can identify any context-free language in the limit.

Lemma 1 (from [23]). *All context-free languages can be described by a grammar in CNF.*

Algorithm 1 ConstructAllGrammars(O)

{O = a set of positive training examples.}
Let $\Sigma = \{b|abc \in O\}$ {i.e. the set of all terminal symbols in O.}
Let $N = \{N_i|1 \leq i \leq (((\sum_j (2 \times |O_j| - 2)) + 1)\} \cup \{S\}$ {i.e. a set of non-terminals.}
Let $\Pi = \{S \rightarrow \varepsilon\} \cup \{A \rightarrow BC|A, B, C \in N\} \cup \{A \rightarrow a|A \in N, a \in \Sigma)$ {i.e. the set of all possible CNF rules using Σ and N.}
Let $G = (N, \Sigma, \Pi, S)$ {i.e. a CNF grammar with all possible CNF rules using Σ and N.}
Let $H = \{(N_i, \Sigma, \Pi_i, S)|\Pi_i \subseteq G\}$ {i.e. all possible sub languages of G.}
return H

Algorithm 2 PruneGrammars(O^+, O^-)

{O^+ = a set of positive training examples and O^- = a set of negative training examples}
Let H = *ConstructAllGrammars*(O^+)
Eliminate all grammars in H that are inconsistent with O^+ and O^-
return H

Algorithm 3 BruteForceLearner

Let $H = \{(\{S\}, \emptyset, \{S \rightarrow \varepsilon\}, S)\}$.
loop
  Randomly select one of the remaining grammars in H as being the hypothesis grammar.
  Get the next sample O_n.
  if O_n is a positive example **then**
    Let $O^+ = O^+ \cup \{O_n\}$.
  end if
  if O_n is a negative example **then**
    Let $O^- = O^- \cup \{O_n\}$.
  end if
  Eliminate all grammars in H that are inconsistent with O^+ and O^-.
  if the $|H| = 0$ **then**
    Let H = *PruneGrammars*(O^+, O^-).
  end if
end loop

Definition 2 (redundant rules). *A grammar $G = (N, \Sigma, \Pi, S)$ contains redundant rules if any rule can be removed without affecting the language generated by G.*

Lemma 2. *Let $G = (N, \Sigma, \Pi, S)$ be a CNF grammar without redundant rules. CalcDerivations(G) (as shown in Algorithm 5) returns a finite set of derivations D with each unique derivation $d_i =$ "$S \Rightarrow^* \mathbf{w}$" $\in D$ deriving a unique sentence $\mathbf{w}$, such that each rule of G is used at least once in D.*

Proof. As listed in the comments of Algorithm 5 we know that $\mathbf{w}$ exists because if it did not exist then we could remove the rule R from G and all sentences could still be be derived. This would imply that R was a redundant rule. It can be seen that $|O| \leq |\Pi|$ and $|D| = |O|$. Therefore both D and O are finite. Each sentence in O has a unique derivation in D, and every rule in Π is used at least once in D. □

Algorithm 4 CalcObservations(G)

Require: $G = (N, \Sigma, \Pi, S)$ is a CNF grammar with no redundant rules.
Let $O = \emptyset$
for all $R \in \Pi$ **do**
  construct a sentence $\mathbf{w}$ such that every derivation tree of $\mathbf{w}$ uses R. {We know that $\mathbf{w}$ exists because if it did not exist then we could remove the rule R from G and all sentences could still be be derived. This would imply that R was a redundant rule.}
  Let $O = O \cup \{\mathbf{w}\}$.
end for
return O

Algorithm 5 CalcDerivations(G)

Require: $G = (N, \Sigma, \Pi, S)$ is a CNF grammar with no redundant rules.
Let $D = \emptyset$
Let O = CalcObservations(G)
for all $\mathbf{w} \in O(G)$ **do**
  Add a single derivation of the form "$S \Rightarrow^* \mathbf{w}$" to D
end for
return D

Lemma 3. *Let $G = (N, \Sigma, \Pi, S)$ be a CNF grammar with no redundant rules. The maximum number of unique non-terminals that can exist in CalcDerivations(G) is defined by the following equation.*

$$\sum_j (2 \times |O_j| - 2)) + 1) \quad (1)$$

Proof. According to [24] for any CNF grammar all derivation trees of strings of length n have $2 \times n - 1$ interior nodes. The non-terminal attached to the root is known to be the start symbol (S). The maximum number of unique non-terminals occurs when all other non-terminals differ, which corresponds to $2 \times n - 2$ non-terminals other than the start symbol for each derivation of a sentence of length n. □

Lemma 4. *Let $\hat{L}$ be a context-free language. Let O = CalcObservationsGivenLang($\hat{L}$) (described in Algorithm 6).*
Claims.

1. *$|O|$ is finite.*
2. *$O \subseteq \hat{L}$.*
3. *When O is presented to the BruteForceLearner algorithm (labelled as positive examples) the algorithm constructs a finite set of hypotheses languages $\hat{H}$ such that $\hat{L} \in \hat{H}$.*
4. *After receiving O the BruteForceLearner algorithm will not construct an alternative set of hypotheses languages.*

Proof. According to Lemma 2 *CalcObservations* is finite so *CalcObservationsGivenLang* is finite. It can be seen from the definition of *CalcObservations* that $O \subseteq \hat{L}$. According to Lemma 3 the maximum number of non-terminals that are required to derive *CalcObservations*(G) is given by Equation 1. By definition it can be seen that the function *ConstructAllGrammars* constructs all possible CNF grammars using the terminal alphabet of *CalcObservations*(G) and a set of unique terminals N where $|N|$ is described by equation 1. Therefore there is at least one grammar $\bar{G}$ in the set H constructed by the function *ConstructAllGrammars* that is identical to G up to the renaming of non-terminals. Therefore there is at least one language $\bar{G}$ in the set of hypotheses languages that is identical to $\hat{L}$. All subsequent positive examples observed by the algorithm will be consistent with $\bar{G}$, and all subsequent negative examples observed by the algorithm will be consistent with $\bar{G}$ therefore the set H will always include $\hat{L}$. Therefore the *BruteForceLearner* algorithm will not construct an alternative set of hypotheses languages.

□

Algorithm 6 CalcObservationsGivenLang($\hat{L}$)

Require: $\hat{L}$ is a context-free language.
Let $G = (N, \Sigma, \Pi, \mathrm{S})$ be a CNF grammar with no redundant rules such that $\mathrm{L}(G) = \hat{L}$. {According to Lemma 1 we know that G exists.}
Remove all redundant rules from G.
return *CalcObservations*(G).

Lemma 5. *Let $\hat{L}$ be a context-free language. Consider the scenario where the BruteForceLearner algorithm is presented with sentences marked as positive or negative according to $\hat{L}$. Let $\hat{H}$ be a finite set of hypotheses languages such that $\hat{L} \in \hat{H}$. Let the BruteForceLearner algorithm contain $\hat{H}$ as its set of hypotheses languages. Let O = CalcAdditional(H, $\hat{L}$) as defined in Algorithm 7.*
Claims.

1. *$|O|$ is finite.*
2. *Items in O are labelled as positive or negative according to $\hat{L}$.*
3. *If the BruteForceLearner algorithm subsequently receives O it eliminates all incorrect hypotheses languages from it's set of hypotheses languages.*
4. *After receiving O the BruteForceLearner algorithm will not construct an alternative set of hypotheses languages.*

Proof. From the definition of *CalcAdditional* it can be seen that the number of samples added to O is less than the number of grammars in H. $|H|$ is finite therefore $|O|$ is finite. From the definition of *CalcAdditional* it can be seen that all samples added to O are labelled as positive or negative according to $\hat{L}$. For all $h_i \in H$ one of the two following scenarios apply.

- $\mathrm{L}(h_i) = \hat{L}$ or
- $\mathrm{L}(h_i) \neq \hat{L}$

Case 1. $\mathrm{L}(h_i) = \hat{L}$
In this case all subsequent positive examples observed by the algorithm will be consistent with h_i, and all subsequent negative examples observed by the algorithm will be consistent with h_i, therefore the grammar h_i will not be deleted from the set H.

Case2. $\mathrm{L}(h_i) \neq \hat{L}$
In this case one of the two following scenarios apply.

– $\mathrm{L}(h_i) \subset \hat{L}$ or
– $\mathrm{L}(h_i) \not\subset \hat{L}$

Case 2.1 $\mathrm{L}(h_i) \subset \hat{L}$

In this case from the definition of *CalcAdditional* it is known that O contains a sentence $\mathbf{w}$ such that $\mathbf{w} \notin \mathrm{L}(h_i)$ marked as a positive example. On receipt of this sentence the *BruteForceLearner* algorithm will eliminate h_i from the set of hypotheses grammars.

Case 2.2 $\mathrm{L}(h_i) \not\subset \hat{L}$

In this case from the definition of *CalcAdditional* it is known that O contains a sentence $\mathbf{w}$ such that $\mathbf{w} \in \mathrm{L}(h_i)$ marked as a negative example. On receipt of this sentence the *BruteForceLearner* algorithm will eliminate h_i from the set of hypotheses grammars.

It can be seen that the cases above cover all possibilities, and in all possibilities for any grammar h_i such that $\mathrm{L}(h_i) \neq \hat{L}$ there exists a sentence in O that causes the *BruteForceLearner* algorithm to eliminate h_i from the set of hypotheses grammars. In all other cases $\mathrm{L}(h_i) = \hat{L}$ and the *BruteForceLearner* algorithm will never eliminate h_i from the set of hypotheses grammars. □

Algorithm 7 CalcAdditional(H, $\hat{L}$)

Require: H is a finite set of context-free languages, $\hat{L}$ is a context-free language.
Let $O = \emptyset$
for all $h_i \in H$ **do**
  if $\mathrm{L}(h_i) \neq \hat{L}$ **then**
    if $\mathrm{L}(h_i) \subset \hat{L}$ **then**
      Let $\mathbf{w}$ be a sentence such that $\mathbf{w} \in \hat{L}$ but $\mathbf{w} \notin \mathrm{L}(h_i)$. {Due to the definition of $\subset$ it is known that such a sentence exists.}
      Add $\mathbf{w}$ to the set O marked as a positive example.
    else if $\mathrm{L}(h_i) \not\subset \hat{L}$ **then**
      Let $\mathbf{w}$ be a sentence such that $\mathbf{w} \in \mathrm{L}(h_i)$ but $\mathbf{w} \notin \hat{L}$. {Due to the definition of $\not\subset$ it is known that such a sentence exists.}
      Add $\mathbf{w}$ to the set O marked as a negative example.
    end if
  end if
end for
return O

Theorem 6. *Let $\hat{L}$ be a context-free language. Let O = CalcCharacteristic($\hat{L}$) as defined in Algorithm 8.*

Claims.

1. *$|O|$ is finite.*
2. *Items in O are labelled as positive or negative according to $\hat{L}$.*
3. *When O is presented to the BruteForceLearner algorithm, the algorithm identifies $\hat{L}$ exactly.*
4. *After receiving O the BruteForceLearner algorithm will not construct an alternative set of hypotheses languages.*

Proof. According to Lemma 4 after the *BruteForceLearner* algorithm receives the sentences in O constructed by *CalcObservationsGivenLang* the *BruteForceLearner* algorithm constructs a set of hypotheses languages $\hat{H}$ such that $\hat{L} \subseteq \hat{H}$. This set of hypotheses is the same set of hypotheses passed to the function *CalcAdditional* in the function *CalcCharacteristic*. Therefore according to Lemma 5 after receiving the remaining sentences in O the *BruteForceLearner* algorithm eliminates all incorrect hypotheses languages from its set of hypotheses languages, and will not construct an alternative set of hypotheses languages. The fact that $|O|$ is finite, and items in O are labelled according to $\hat{L}$ can derived from the definition of *CalcCharacteristic*, Lemma 4 and Lemma 5. □

Algorithm 8 CalcCharacteristic($\hat{L}$)

Require: $\hat{L}$ is a context-free language.
Let $O^+ = CalcObservationsGivenLang(\hat{L})$
Let $O^- = \{\ \}$
Let $H = PruneGrammars(O^+, O^-)$
Let $C = O + CalcAdditional(H, \hat{L})$ {Note C is an ordered list.}
return C

Theorem 7. *Let $\hat{L}$ be a context-free language. Let O = CalcObservationsGivenLang($\hat{L}$). Let C = CalcCharacteristic($\hat{L}$).*

Claim: *On receipt of C the total number of hypotheses grammars considered by the algorithm is described by the following equation where $T(O)$ is the set of terminal symbols in O.*

$$2^{1+(((\sum_j (2\times|O_j|-2))+1)^3+((\sum_j (2\times|O_j|-2)+1)T(O)))} \quad (2)$$

Proof. From the definition of *ConstructAllGrammars* it can be seen that the number of hypotheses grammars constructed by the function when presented with O is given by Equation 2. It can also be seen that if $Q_1 \subseteq Q_2$ then $\mathit{ConstructAllGrammars}(Q_1) \subseteq \mathit{ConstructAllGrammars}(Q_2)$. From the definition of the *BruteForceLearner* algorithm it can be seen that although the function *ConstructAllGrammars* may be called numerous times, each time that it is called, it is called with a superset of the sentences that it was called with previously. According to Theorem 6 the last time that the function *ConstructAllGrammars* will be called, it will be called with O. Therefore Equation 2 describes the total number of hypotheses grammars considered by the *BruteForceLearner* algorithm. □

Lemma 8. *If the BruteForceLearner algorithm correctly identifies a context-free language it does so in time exponential with the size of the target language.*

Proof. Let S_1 be the the number of rules of the smallest CNF context-free grammar that can be constructed to represent the target language. Let S_1 be a measure of the size of the target language. Let T be the time taken by the *BruteForceLearner* algorithm to correctly identify the target language. Let S_2 be the number of candidate rules that the function *ConstructAllGrammars* is considering when constructing its set of hypotheses grammars. If the algorithm has correctly identified the target language then $S_2 \geq S_1$. The number of grammars created by the function *ConstructAllGrammars* is 2^{S_2}. If we assume that the time taken to construct each one of these grammars is t_2 then $T > t_2 \times 2^{S_2} \geq t_2 \times 2^{S_1}$. □

At this point we have a method of constructing a set of positive and negative examples from any context-free language, such that we know that it is possible to infer the language exactly from those training examples (although not necessarily in polynomial time). We also know the size of the hypothesis space that would be considered by at least one grammatical inference algorithm that can do this, and the size of this hypothesis space can be used as a metric of the complexity of the grammatical inference task.

If the target language is described using a deterministic grammar in CNF, we know that it does not contain any redundant rules, and we know that all derivations of any sentence generated by a given rule R must use R. This is known because each sentence has only one derivation. Therefore given any deterministic context-free grammar G in CNF, we can construct a set of sentences equivalent to $\mathit{CalcObservations}(G)$ simply by generating for each rule in G one sentence derived using that rule. It can be shown by constructive proof that this technique can be used even if G is not in CNF. Therefore we have a way of constructing $\mathit{CalcObservations}(G)$ in polynomial time with respect to the number of rules in the target grammar. Therefore this technique was used in the Omphalos competition to construct a portion of the training sentences from each deterministic target grammar. A similar technique was also used to construct a set equivalent to $\mathit{CalcObservations}(G)$ from those non-deterministic grammars used in the competition. This was made possible because the non-deterministic grammars were created by adding non-deterministic constructs to deterministic grammars.

The technique for constructing the set of sentences $\mathit{CalcCharacteristic}(\hat{L})$ described in the proofs however cannot be executed in polynomial time. This is because the function *PruneGrammars* is used which cannot be executed in polynomial time. Therefore if the complexity of the learning tasks was small enough to enable the set $\mathit{CalcCharacteristic}(\hat{L})$ to be constructed, then the learning tasks would be too easy. Instead we made use of the following theorem that can be informally interpreted to mean that on receipt of a "sufficiently" large enough set of unique sentences tagged according to a target language $\hat{L}$ the *BruteForceLearner* will infer $\hat{L}$ exactly. This theorem can also be formally stated as "the *BruteForceLearner* can identify any context-free language in the limit from positive and negative data" [1].

Theorem 9. *On receipt of an infinite stream of unique sentences correctly labelled as being either an element or not an element of a target context-free language $\hat{L}$, at some finite time the BruteForceLearner algorithm will contain a set of hypotheses grammars H such that for all $h_i \in H$: $L(h_i) = \hat{L}$, and after that point the algorithm will not change its hypotheses.*

Proof. According to Lemma 4 there exists at least one finite set of sentences $O \subseteq \hat{L}$ labelled as pos-

itive examples that when presented to the *BruteForceLearner* algorithm, will cause it construct a set of hypotheses languages $\hat{H}$ such that $\hat{L} \subseteq \hat{H}$, and upon receipt of O the *BruteForceLearner* algorithm will not construct an alternative set of hypotheses languages. Therefore at some finite point in time the algorithm will receive a set of sentences O_4 such that $O \subseteq O_4$. At that point the algorithm will construct a set of hypotheses languages $\hat{H}$ such that $\hat{L} \subseteq \hat{H}$, and upon receipt of these sentences the algorithm will not construct an alternative set of hypotheses grammars.

According to Lemma 5 there exists at least one additional set of positive and negative training sentences O_2 that when presented to the *BruteForceLearner* algorithm after the presentation of O_4 cause it eliminate all incorrect hypotheses languages from it's set of hypotheses languages. Therefore at some finite point in time the algorithm will observe a set of positive and negative sentences O_5 such that $O_5 \subseteq O_2$ and after that time the algorithm will have eliminated all incorrect hypotheses in its set of hypotheses grammars, and after that point it will not change its set of hypotheses. □

3.1. Updated Theory

Theorem 9 would be more useful if it was known how many more additional samples are needed to be created, so that it can be guaranteed that the problems can be solved by at least one grammatical inference algorithm. At the time of the creation of the training sets for the Omphalos competition, there was no known procedure for determining how many samples in addition to the set *CalcObservations* should be produced. At the time of setting the competition tasks a decision was made to make the number of additional samples to be between 10 and 20 times the number of sentences in the set *CalcObservations*(G). This number was somewhat arbitrary based upon past experience and gut feel ("instinct"). In hindsight it appears to have been sufficient.

Since the completion of the competition however some improvements have made to the theory, and we can now generate a conservative estimate of the number of sentences that should be created for each competition task, using PAC learning theory. After the receipt of the set *CalcObservations*(G) the *BruteForceLearner* algorithm has a finite hypothesis space of known size. Therefore we can use the following equation that provides a upper bound on the number of additional labelled samples (m) sufficient for any consistent learner (and therefore the *BruteForceLearner*) to probably (with probability $1-\delta$) approximately (within error ε) infer the correct language [25].

$$m \geq \frac{1}{\varepsilon}(\ln|H| + \ln(1/\delta)) \tag{3}$$

The number of samples returned by the Equation 3 using Equation 2 to calculate $|H|$ is very large indeed. For instance if Equation 3 is used with $|H|$ defined using Equation 2 for the simplest grammar in the competition with $|H| = 2^{1.1\times10^9}$ and $\varepsilon, \delta = 0.01$ at least 7.62×10^{10} examples in addition to the 16 examples in O would be required (The target grammar had 23 rules). In actuality the problem was solved by three different competitors with only 266 positive and 802 negative examples.

As a result of this the *BruteForceLearner* was examined again to see if the size of the hypothesis space could be drastically reduced, while preserving the identification in the limit properties. The intuition that lead to the definition of the *BruteForceLearner* algorithm was that given a set of sentences like the one generated by function *CalcObservations*, there are a finite number of unlabelled derivation trees that can be assigned to those sentences, and a finite number of maximum non-terminals that could be used by those derivations. On receipt of these sentences however the *BruteForceLearner* algorithm would not know when it had received a superset of *CalcObservations*, but would rather take a guess at the number of non-terminals required, and then only reconstruct a larger hypothesis set, when no grammar in the hypothesis set was consistent with the training data. An alternative strategy is to use a variable $\hat{n}$ to denote a guess of the number of non-terminals in the target grammar. Whenever a new hypothesis set needs to be constructed the algorithm could just increment $\hat{n}$ and construct all CNF grammars with exactly $\hat{n}$ non-terminals. With a slight modification to the proofs listed earlier in this document it can be seen that the algorithm will at some point, create a hypothesis space that is sufficient while consider the following number of hypothesis grammars.

$$2^{1+n^3+n\times T(O)} \tag{4}$$

where n is the number of non-terminals in the smallest CNF grammar that can represent the target language. The set of additional sentences could then be created using the function *CalcAdditional* using the smaller set of hypothesis grammars. It should be noted that in this case not only would creating this number of grammars be sufficient, but that the target language could not be identified with a smaller hypothesis of the number of non-terminals than n. For this reason Equation 4 is a better measure of the complexity of the grammatical inference task than Equation 2. For the sake of consistency however we will continue to use Equation 2 in this paper as the measure of the complexity of a grammatical inference task, although in future competitions Equation 4 may be used.

It is however still possible for the *BruteForceLearner* to consider an even smaller set of hypothesis grammars, and still identify all context-free grammars in the limit. Constructing this smaller set however would most likely result in even longer execution times however, but considering this smaller hypothesis set is of use in determining the number of additional samples that should be constructed to pass to the *BruteForceLearner* algorithm in addition to those constructed using *CalcObservations*. The *BruteForceLearner* algorithm can consider a smaller set of grammars because the Chomsky normal form is not a unique normal form. Therefore many of the grammars in the set of hypothesis grammars will describe the same language. This can lead to problems even when using Equations 3 and 4 to estimate the number of additional samples required. This is because the value returned by Equation 3 is close to the lower bound on the number of additional labelled samples required only when the number of incorrect hypotheses in H is close to $|H|$. This is only the case when every grammar in H describes a unique language, which is not the case. For instance consider any grammar in H that has r non-terminals (including the start symbol) out of a possible n non-terminals (including the start symbol). In this case the number of other grammars that are isomorphic (identical up to the renaming of non-terminals) to this grammar in H would be the number of permutations of $(n-1)$ in $(r-1)-1$, i.e. $((n-1)!/(n-1-(r-1))!)-1 = ((n-1)!/(n-r)!)-1$. In addition, because the CNF is not a unique normal form, many of the grammars will describe the same language even if they are not isomorphic. For instance all grammars that contained useless rules will describe the same language as the grammar with those useless rules removed (which will also exist in H). Also many instances of left recursion can be replaced by an instance of right recursion without changing the language described by the grammar. The hypothesis space will also include some languages that cannot generate the training examples from which the hypotheses space was created.

As an alternative to using the *ConstructAllGrammars* described early, the *BruteForceLearner* algorithm could construct a set of hypothesis grammars in which no grammar is isomorphic to another, the start symbol exists in every grammar, and for all terminal symbols a there is at least one rule of the form N $\rightarrow$ a. It can be shown that the size of this set of grammars is $\sum_{n=1}^{N}\sum_{r=1}^{n}\frac{r(2^{r^2}-1)\times T(2^r-1)\times 2}{(r-1)!}$. This equation can be approximated using the following equation.

$$2^{\log_2 N+N^2+\log_2 T+N+1-\sum_{x=1}^{N-1}\log_2 x} \quad (5)$$

Table 1 gives the number of estimated additional samples required using Equations 3 and 4 and Equations 3 and 5. The table also gives an indication of the reduction in $|H|$ using the different techniques described in this section (i.e. columns 2 to 5).

4. Constructing the Competition Tasks

This section of the document describes the mechanism by which the competition tasks were constructed. First we will discuss the way in which the complexity of the competition tasks were chosen. Then the method by which grammars were constructed will be discussed, followed by the way in which the training and testing examples were created.

4.1. Setting the complexity of the competition tasks

As one of the major motivations of the Omphalos competition was to promote the development of new grammatical inference algorithms, it was desirable that the learning tasks in the competi-

Table 1
Estimates of required number of additional training sentences using different equations

Problem	Complexity			Size of Training Set	Theoretical Size Required	
	Eq 2	Eq 4	Eq 5		Eq 4 & 3	Eq 5 & 3
2 & 6.2	7.12×10^8	2.54×10^5	3.76×10^3	280	7.66×10^6	113,239
4 & 6.4	1.13×10^{10}	5.68×10^6	3.08×10^4	418	1.71×10^8	927,348

tion were sufficiently hard so as to be just outside the state-of-the-art. Ideally, the best of the current systems should be able to solve the simplest problems, but not the more difficult ones. Omphalos should give an idea of where the current state-of-the-art currently is, and try to push the boundary from there.

As discussed earlier in this paper, Equation 2 was considered to be a suitable measure of the complexity of a grammatical inference task. To apply this equation one begins with a grammar defining the target language. Then a set of positive examples is generated from the grammar using a function that approximates the function *CalcObservations*. Then the number of terminal symbols in this set is calculated, and finally Equation 2 is applied.

In order to estimate the complexity of problems that could be solved by state-of-the-art algorithms the literature was reviewed to identify pre-existing benchmark grammatical inference problems. The work in [26] and [27] identified some benchmark problems, i.e. grammars that can be used as some sort of standard to test the effectiveness of a GI system by trying to learn these grammars. The grammars were taken from [28] and [24]. Using Equation 2 the complexities of these grammars were calculated. A description of these grammars and their complexity measures are listed in Table 2. Using these results the complexities of the target grammars of the competition problems were selected. Once again based upon "gut feel" it was decided to set the complexity of the easiest competition problem at around 2^{10^9} despite the fact that the complexity of the most difficult problem listed in Table 2 was around 2^{10^5}.

Table 3 lists the complexities of the Omphalos competition problems. It can be seen that the competition included several grammatical inference tasks that were ranked according to difficulty. The ranking of these problems was based upon three axes of difficulty in grammatical inference, specifically:

1. The complexity of the underlying grammar (as defined by Equation 2),
2. whether or not negative data is available and,
3. how similar the negative examples in the test set are to positive examples of the language. For instance whether or not the test set includes sentences that can be generated by regular approximations to the target language but not the target language itself.

At the close of the competition the competitor that had solved the highest ranking problem was declared the winner of the competition overall.

The reason why the complexity measure defined using Equation 2 was not used as the sole metric for ranking the problems was to encourage participation from as many researchers as possible. For instance the ranking methodology enabled algorithms that used positive data only, to compete with algorithms that used both positive and negative data. Specifically, the successful inference of a target language from positive data only was ranked higher than the successful inference of a language from positive and negative data, all other things being equal. For instance, problem 2 is ranked higher than problem 1.

During the lifetime of the competition the third axis of difficulty was introduced when it became clear that some of the original test data did not accurately determine whether or a competitor had inferred the target language or a regular approximation to it. Correctly labelling a test set in which the negative sentences closely resembled the positive sentences was ranked higher than correctly labelling a test set where the negative examples differed greatly from the positive examples. For instance, problem 6.1 is ranked higher than problem 1 and even problem 6.

Table 2
Benchmark grammatical inference problems.

	Description	Example phrase	$\log_2$ compl.	Properties
1	(aa)∗	aa, aaaa, aaaaaa	1.34×10^3	Regular
2	(ab)∗	ab, abab, ababab	2.22×10^3	Regular
3	Operator precedence (small)	(a+(b+a))	9.37×10^3	Not regular
4	Parentheses	(), (()), ()(), ()(())	2.22×10^3	Not regular
5	English verb agreement (small)	that is a woman, i am there	5.33×10^5	Finite
6	English lzero grammar	a circle touches a square, a square is below a triangle	4.17×10^6	Finite
7	English with clauses (small)	a cat saw a mouse that saw a cat	6.59×10^5	Not regular
8	English conjugations (small)	the big old cat heard the mouse	9.13×10^5	Regular
9	Regular expressions	ab∗(a)∗	9.39×10^3	Not regular
10	$\{\omega = \omega^R\ ,\ \omega \in$ {a,b}{a,b}+$\}$	aaa, ba	6.90×10^4	Not regular
11	Number of a's=number of b's	aabbaa	4.29×10^4	Not regular
12	Number of a's=2×number of b's	aab, babaaa	3.01×10^5	Not regular
13	$\{\omega\omega \mid \omega \in$ {a,b}+$\}$	aba, aa	9.12×10^4	Regular
14	Palindrome with end delimiter	aabb$, ab$, baab$	1.18×10^5	Not regular
15	Palindrome with center mark	aca, abcba	4.96×10^3	Not regular
16	Even length palindrome	aa, abba	9.30×10^3	Not regular
17	Shape grammar	da, bada	2.45×10^4	Not regular

All other things being equal, solving a problem with a higher complexity measure was ranked higher than solving one with a lower complexity measure. For instance, problem 3 is ranked higher than problem 1.

In addition it was decided to make the highest ranking problems non-deterministic languages, while the lowest ranking problems were deterministic languages. There were two reasons for this. Firstly it was been noted by [27] that the inference of non-deterministic languages is a more difficult task than the inference of deterministic languages. Therefore solving those problems that involved non-deterministic languages was ranked higher than solving those problems that involved deterministic languages. Secondly deterministic languages form a useful subset of context-free languages that are a superset of regular languages. There is a large amount of background knowledge on deterministic languages, that might useful for the inference of deterministic languages, but at the time of the setting of the competition was not being applied to the task of grammatical inference. It was hoped that the focus on the inference of deterministic context-free languages might be a catalyst for advances in grammatical inference.

Lastly it should be noted that the Omphalos competition employed an interactive technique in setting the complexity of the grammatical inference tasks. Although an initial set of six problems were created, the competition was arranged so that if these problems were solved early in the life of the competition, then additional more difficult problems could be added. Similarly, if after an extended amount of time the competition tasks were not solved, easier problems could be added to the list of competition tasks. By the setting of the difficulty of the competition tasks using this technique, we could get a feel of the complexity of grammatical inference problems that could be solved by state-of-the-art algorithms.

4.2. Creation of the Target Grammars

After examining the complexities of those languages listed in Table 2 it was decided that the complexity of the easiest tasks (tasks 1 & 2) should be in the order of 2^{10^9}. This complexity measure was chosen for three reasons. Firstly it was a few orders of magnitude higher than those tasks listed in Table 2. Secondly problems of this complexity could not be learnt using a brute force tech-

Table 3
Complexities of Benchmark Inference Problems in Omphalos Competition.

	Training data	Properties	$\log_2$ compl.
1	Positive and negative	Not regular, deterministic	1.10×10^9
2	Positive only	Not regular, deterministic	7.12×10^8
3	Positive and negative	Not regular, deterministic	1.65×10^{10}
4	Positive only	Not regular, deterministic	1.13×10^{10}
5	Positive and negative	Not regular, non-deterministic	5.46×10^{10}
6	Positive only	Not regular, non-deterministic	6.55×10^{10}
6.1	Positive and negative	Not regular, deterministic	1.10×10^9
6.2	Positive only	Not regular, deterministic	7.12×10^8
6.3	Positive and negative	Not regular, deterministic	1.65×10^{10}
6.4	Positive only	Not regular, deterministic	1.13×10^{10}
6.5	Positive and negative	Not regular, non-deterministic	5.46×10^{10}
6.6	Positive only	Not regular, non-deterministic	6.55×10^{10}
7	Positive and negative	Not regular, deterministic	5.88×10^{11}
8	Positive only	Not regular, deterministic	1.63×10^{11}
9	Positive and negative	Not regular, non-deterministic	1.08×10^{12}
10	Positive only	Not regular, non-deterministic	9.92×10^{11}

nique (such as the *BruteForceLearner* algorithm) in time to be entered into the Omphalos competition. Thirdly based upon published execution times of existing algorithms it was estimated that these problems could not be solved using existing published techniques in time to be entered into the competition. This decision was made to encourage the development of new grammatical inference techniques.

It was decided that higher ranking problems should have a complexity measure that was at least an order greater than similar tasks lower in the ordering to ensure that they truly were more difficult tasks. Once the form and complexities of the target grammars were decided upon, the grammars were created as follows:

1. The number of non-terminals, terminals and rules were estimated based upon the number of non-terminals, terminals and rules in the benchmark problems (Table 2).
2. A set of terminals and non-terminals were created. Rules were then created by randomly selecting terminals and non-terminals. A fixed number of rules were created to contain only terminal strings.
3. Useless rules were then identified. If a non-terminal could not generate a terminal string, a terminal rule was added to it. If a non-terminal was not reachable from the start symbol, rules were added to ensure the rule was reachable from the start symbol. For instance if the non-terminal N was unreachable from the start symbol, a rule was created with the start symbol on the left hand side of the rule, and N on the right hand side of the rule.
4. Additional rules were added to ensure that the grammar did not represent a regular language. Specifically rules containing center recursion were added.
5. The complexity of the grammar was calculated and if it was not in the desired range, then the grammar was deleted.

Tests were then undertaken to ensure that grammars 1–4 represented deterministic languages. Specifically $LR(1)$ parse tables were constructed from the grammars using Bison [29]. To ensure that grammars 4 and 5 represented non-deterministic languages, rules were added to the target grammars. After problem 6 was solved, problems 7 to 10 were added to the competition. Problems 6.1 to 6.6 were added later in the competition when it appeared that problems 7 to 10 would not be solved during the life of the competition.

4.3. Construction of training and testing sets.

Once the target grammars were constructed, sets equivalent to *CalcObservations* were constructed for each grammar. Sets of positive examples were then created using the GenRGenS software [30].

For the first six problems additional examples were then created by randomly generating sentences of length up to five symbols more than the length of the longest sentence in *CalcObservations*(G). These sentences were then parsed using the target grammar and were labelled as being either in or out of the target language. This set of sentences was then randomized and split into testing and training sets, but in such a way as to ensure that the training set contained a superset of *CalcObservations*(G). For those problems that were to be learnt from positive data only the training sets had all negative examples removed.

For problems 6.1 to 10 a more rigorous method of constructing negative data was used as follows:

- For each context-free grammar an equivalent regular grammar was constructed using the superset approximation method based on Recursive Transition Network (RTN) described in [31]. Sentences that could be generated from this regular approximation to the target language were included as negative data. These sentences were included to distinguish between competitors who had created regular approximations to the underlying context-free languages, and competitors who had identified a non-regular language.
- A context-free grammar that defined a grammar that was a superset of the the target language was constructed by treating the target grammar as a string rewriting system, and normalizing the right hand sides of rules using the normalization algorithm described in [32]. That is, a context-free grammar was constructed that described a language that was a superset of the target language, but in which the right hand side of each rule could not be parsed by the right hand side of any other rule. Negative examples were then created from this approximation to the target language.
- Each target grammar in the competition included some deliberate constructs designed to trick grammatical inference algorithms. For instance most included sequences that were identical to center recursion expanded to a finite depth. An example is $a^n b^n$ where $n < m$, m is an integer > 1. To ensure that the training and testing examples tested the ability of the inferred grammars to capture these nuances, the target grammars were copied and hand modified changing the $a^n b^n$ where $n < m$ to become $a^n b^n$ where $n > 1$. In addition, where center recursion existed of the form $a^n b^n$ in the target grammar the regular approximations $a^* b^*$ were included in the "tricky" approximation to the target grammar. Negative examples were then created from these approximations to the target grammar and added to the test set.
- There were an equal number of positive and negative examples in the test sets.

In addition for problems 6.1 to 10:

- The longest training example was shorter than the longest test example.
- The grammar rules for problems 7 to 10 were shorter than for problems 1 to 6.6 and had more recursion. Some non-$LL(1)$ constructs were also added to the target grammars for problems 7 to 10.

4.4. Competition Infrastructure

The Omphalos competition reused a number of pieces of software from the Abbadingo competition [20] and the Gowachin server, including the Oracle that was consulted by competitors to determine if they had successfully solved the competition tasks or not. To enter the competition, competitors needed to:

1. Download the training data and test data from http://www.irisa.fr/Omphalos/.
2. Train their system using the training data.
3. Apply their system to the test data, resulting in a list of 1's and 0's, indicating that each sentence in the test set is either in or is not in the language (respectively).
4. Submit the list of 1's and 0's to the Omphalos "Oracle" using an html form on the web-site.
5. The Oracle would then notify the competitor, that either they were successful or unsuccessful.

The decision to provide only a pass or fail to competitors was a deliberate design choice, to prevent that information to being used by competitors for Hill Climbing (test set tuning). In addition if any competitor had deliberately consulted the Oracle more than 25 times in one day they would have been disqualified.

5. Results of Omphalos

The Omphalos competition ran for eight months in 2004 from February the 5^{th} until October 11^{th}. During that time the competition data was downloaded from seventy different internet domains, on almost all continents.

The web-site itself received over 1000 visits from over 270 domains.[6] There were 139 label submissions by 8 contestants to the Oracle. Of the 16 competition problems, 8 were solved during the life of the competition, and one other was solved after the competition had closed. Most importantly the winner of the competition developed some new techniques that have pushed the state-of-the-art in grammatical inference. Table 4 contains details of key dates in the competition. Alexander Clark was declared the winner of the competition on October 1^{st}. A detailed journal article on his winning techniques has been submitted [33].

If the number of different domains from which the competition data was downloaded from is indicative of the number of competitors, then the number of competitors (70) is almost 50% more than the number of people who attended ICGI 2004, and greater than the number of estimated individuals who downloaded the Abbadingo competition data. The number of individuals who actually submitted submitted labellings to the Oracle however was extremely low. From discussions during the Omphalos session at ICGI 2004, email correspondence and from published comments [34] the reason for this rate of submission was two fold. Predominantly many competitors stated that they run their existing grammatical inference systems on the competition data, and using cross validation decided that the performance of their algorithms on this data was too poor to warrant the submission of a test set to the Oracle. Other researchers reported that due to other commitments they were unable to spend enough time on the tasks. It should be noted that all machine learning algorithms contain learning bias, as does the method by which the target grammars were generated. Nevertheless, because of the high participation rate and the complexity of the problems solved we consider that it can be concluded the winning algorithm is world class and the complexity of the solved problems represent the limit of problems that can be solved using state-of-the-art algorithms.

[6] Attempts were made to discard crawlers and bot hits.

5.1. Problems 1, 3, 4, 5, and 6

Problem 1 was solved by Joan Andreu Sánchez from the Departament de Sistemes Informàtics i Computació, Universitat Politècnica de València. Although Sánchez originally tried to solve the problem using the Inside-Outside algorithm, he actually solved it manually. After discovering the regularities in the positive examples he used a regular expression constructed by hand to classify the test examples as being either positive or negative. Although the target grammar was not a regular language the test sets did not include a single non-regular example. This in addition to the speed in which the problem was solved suggests that the first task was overly simple, and the negative examples were too different from the positive examples to be an accurate test of whether or not the language had been successfully learnt.

Problems 3, 4, 5, and 6 were solved by Erik Tjong Kim Sang from the CNTS - Language Technology Group at the University of Antwerp in Belgium. Tjong Kim Sang used a pattern matching system that classifies strings based on n-grams of characters that appear either only in the positive examples or only in the negative examples. With the exception of problem 1 this technique was not sufficient to solve the problem, so Tjong Kim Sang generated his own negative data, using the principle that the majority of randomly generated strings would not be contained within the language. His software behaved as follows;

1. Firstly it loaded positive examples in from the training file.
2. It then generated an equal number of unseen random strings, and added these to the training data as negative examples.

Table 4
Competition time line.

Date	Event
February 15^{th} 2004	Competition begins
February 20^{th} 2004	Problem 1 solved by Joan Andreu Sánchez
March 22^{nd} 2004	Problems 3, 4, 5 and 6 were solved by Erik Tjong Kim Sang
April 14^{th} 2004	Problems 7, 8, 9 and 10 were added
June 7^{th} 2004	New larger testing sets were added for problems 1 to 6
September 6^{th}	Problems 2 and 6.2 were solved by Alexander Clark
September 7^{th}	Problem 6.4 was solved by Alexander Clark
October 1^{st} 2004	Competition closed
October 1^{st} 2004	Alexander Clark was declared the winner of the competition
October 11^{th} 2004	Omphalos session at ICGI 2004

3. A n-gram classifier was then created as follows: A count was made of n-grams of length 2 to 10 that appeared uniquely in the positive examples or uniquely in the negative examples. A weighted (frequency) count of such n-grams in the test strings was then made. For each sentence in the test set. If the positive count was larger than the negative count then the string was classified as positive, otherwise it was classified as negative. If a string contained two zero counts then that sentence was classified as unknown.
4. Steps 2 and 3 were repeated thirty times.
5. Strings that were always classified as positive in all thirty tests were then assumed to be positive examples. Strings that were classified as negative one or more time were classified as negative. Other strings were classified as unknown.

5.2. Problems 6.2 and 6.4

Problems 6.2 and 6.4 were solved by Alexander Clark from the University of Geneva. Clark's approach was based on a simplified version of the algorithm described in [35] that also made use of some of properties of deterministic context-free grammars. Alexander also used some Alignment-based learning techniques [36]. It should be noted that not only did Clark correctly label a large number of similar test sentences (12,004 and 17,230) after training his algorithm from an extremely small number of positive sentences (281 and 419), at the completion of the competition his grammars were compared to the target grammars and it was found that he had identified the target languages exactly. The target grammars had 24 and 38 rules respectively.

Clark's algorithm can best be described as a model merging technique, whereby constituents are firstly identified and assigned arbitrary non-terminals. The constituents and sentences are then reduced using other constituents. This reduction process adds structure to the sentences, and reduces the size of the hypothesis grammar. Non-terminals are then merged, which generalizes the language described by the grammar.

There are two features of Clark's technique that make it distinct from other model merging techniques. The first of these is the use of mutual information to filter candidate constituents [35]. The second is the use of substitution graphs to identify whether or not constituents can be used interchangeably.

Mutual information is an information-theoretically motivated measure for discovering interesting collocations between different events, in this case the collocations of words before and after constituents.

Consider the alphabet $\sum$ augmented with a distinguished symbol #. For a substring $\mathbf{w}$ that occurs in the sentence $a\ \mathbf{w}\ b$ with $a, b \in \Sigma^*$, we can define l to be the final symbol of a if there is one, or to be # if $a = \varepsilon$. Similarly we define r to be the first symbol in b or #.

Let $a(\mathrm{l}, \mathbf{w}, \mathrm{r})$ be the number of times that $\mathbf{w}$ occurs in the context (l, r).

Let $b(\mathrm{l}, \mathbf{w})$ be the number of times l appears before the substring $\mathbf{w}$ i.e. $\sum_{\mathrm{r}} a(\mathrm{l}, \mathbf{w}, \mathrm{r})$.

Let $c(\mathbf{w}, \mathrm{r})$ be the number of times r appears after the substring $\mathbf{w}$ i.e. $\sum_{\mathrm{l}} a(\mathrm{l}, \mathbf{w}, \mathrm{r})$.

Let $d(\mathbf{w})$ be the number of times that the substring $\mathbf{w}$ appears in the training sample i.e. $\sum_{\mathrm{l,r}} a(\mathrm{l}, \mathbf{w}, \mathrm{r})$.

Clark uses these counts in the following equation to calculate mutual information.

$$MI(\mathbf{w}) = \sum_{\mathrm{l}\in\Sigma\cup\{\#\}} \sum_{\mathrm{r}\in\Sigma\cup\{\#\}} \frac{a(\mathrm{l},\mathbf{w},\mathrm{r})}{d(\mathbf{w})} \log \frac{a(\mathrm{l},\mathbf{w},\mathrm{r})d(\mathbf{w})}{b(\mathrm{l},\mathbf{w})c(\mathbf{w},\mathrm{r})}$$

Clark claims that substrings that are constituents will tend to have a high value of this quantity, whereas substrings that are not constituents will have a low value of this quantity.

As an example, the counts of the local contexts of the string "q g m t" in the training set of problem 6.4 are given in Table 5. According to Table 5 for instance the sequence "k q g m t h" appeared 69 times in the training set. From these counts mutual information is calculated as shown in Table 6. It can be seen that the mutual information of the sequence "q g m t" is 1.00.

Table 5
Counts of all contexts of the sequence "q g m t".

	h	k	r
k	69	14	0
n	0	0	75

Table 6
Mutual information of the sequence "q g m t".

l	r	$a(l,\mathbf{w},r)$	$b(l,\mathbf{w})$	$c(\mathbf{w},r)$	$d(\mathbf{w})$	MI
k	h	69	83	69	158	0.41
k	k	14	83	14	158	0.08
n	r	75	75	75	158	0.51
Total						1.00

Whether or not the mutual information is sufficiently high depends upon the setting of a threshold value in the algorithm. In this case the high value of mutual information calculated (1.00) suggests that "q g m t" is a true constituent. Thus Clark's algorithm assigned the non-terminal NT_{16} to it, e.g. $\mathrm{NT}_{16} \Rightarrow^* \mathrm{qgmt}$. As a result the grammar inferred by Clark contains three rules as follows.

$\mathrm{NT}_{16} \rightarrow \mathrm{qgmt}$
$\mathrm{NT}_{14} \rightarrow \mathrm{nNT}_{16}\mathrm{rq}$
$\mathrm{NT}_{4} \rightarrow \mathrm{kNT}_{16}\mathrm{NT}_{4}\mathrm{owNT}_{3}$

These three rules map onto the following three rules in the grammar from which the training examples were generated.

$\mathrm{Y}_{3} \rightarrow \mathrm{qgmt}$
$\mathrm{Y}_{11} \rightarrow \mathrm{nY}_{3}\mathrm{rqY}_{5}\mathrm{xe}$
$\mathrm{Y}_{6} \rightarrow \mathrm{kY}_{3}\mathrm{Y}_{6}\mathrm{owY}_{9}$

The identification of constituents enables the introduction of non-terminals into the hypothesis grammar, but for the grammatical inference algorithm to generalise from the training examples, some assumptions need to be made about the substitutability of constituents. A common guessing strategy used by grammatical inference algorithms is that if $\mathrm{A} \rightarrow^* \mathbf{u}$, $\mathrm{A} \rightarrow^* \mathbf{v}$ and $x\mathbf{u}y \in L$ then it is assumed that $x\mathbf{v}y \in L$. While this holds true for some context-free languages, in general it cannot be assumed. To identify the instances when this guessing strategy should be applied, and when it should not, Clark constructed a data structure that he calls a substitution graph. For each substring $\mathbf{u}$ observed in the training sample a node is constructed in an undirected graph. An arc is then created between two nodes $\mathbf{u}$ and $\mathbf{v}$ (representing the substrings $\mathbf{u}$ and $\mathbf{v}$) if there are strings l and r such that both $l\ \mathbf{u}\ r$ and $l\ \mathbf{v}\ r$ are in the training sample. For instance Figure 3 shows part of such a graph inferred by Clark's algorithm for problem 4. Here we can see that the constituents "q t b u r d b o v", "v k g i o o" and "p i p l d s k" are observed to be fully interchangeable with one another in the training sample. In contrast however if the constituents "q t b u r d b o v" and "v k g i o o" were interchangeable with another and similarly "v k g i o o" and "p i p l d s k" were interchangeable but "q t b u r d b o v" and "p i p l d s k" were not then with a sufficiently large enough training sample a substitution graph similar to that shown in Figure 4 would be constructed.

Clark's technique can be summarized as follows.

- The algorithm calculated all substrings observed in the training sample using a suffix array. All substrings that occurred greater than a predefined number (f_{min}) of times were considered to be candidate constituents.
- The algorithm then discarded all candidate constituents whose mutual information was less than a predefined number (M_{min}).

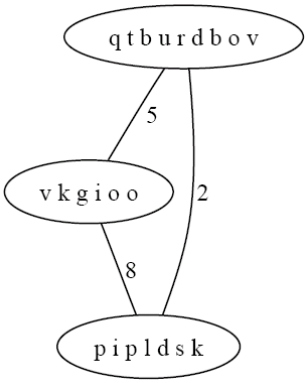

Fig. 3. Part of a substitution graph showing the observed substitutability of constituents (from [33])

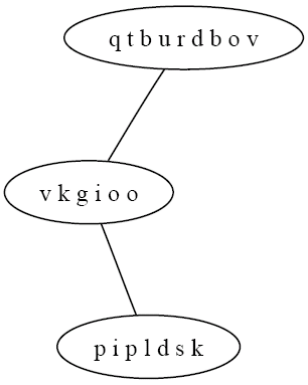

Fig. 4. An alternative substitution graph showing an incomplete subgraph (limited substitutability)

- A substitution graph was then constructed to identify cliques, i.e, which constituents were substitutable with one another.
- Sentences were then reduced using the constituents.

Clark made multiple attempts at those problems that he solved. For instance he made seven attempts at problem 6.2 before solving it. After inferring a grammar from the training data that did not give a correct answer on the test data, Clark assumed that his inferred grammar was too specific and so he generalized it. He did this by implementing a series of heuristic modifications to the hypothesised grammars. Given two rules in the hypothesis grammar of the form A $\rightarrow$ **UWV** and B $\rightarrow$ **W**, he sometimes generalised the first rule by replacing it with A $\rightarrow$ **U**B**V**. Conversely, if a rule existed of the form A $\rightarrow$ **U**B**V** such that for every string in the training sample the occurrence of B in this rule is expanded by a particular rule B $\rightarrow$ **w**, then he sometimes made the grammar more specific by replacing the first rule with A $\rightarrow$ **UwV**. In addition there were a number of free parameters that could be tweaked. Therefore there appears to still be a certain amount of "gut feel" and "tweaking" involved in the solving these problems. This would appear to be a sound strategy for solving complicated tasks like the Omphalos competition, or real world applications. Clark's advice for a successful strategy is to exploit useful properties that randomly generated grammars are likely to have.

Although it is known that all context-free languages cannot be learnt using positive examples only [7] Clark's algorithm used a range of techniques to prevent the algorithm from overgeneralizing as summarized below.

- The context distributions needed to be similar using a local measure of similarity (MI).
- There needed to exist two strings c and d such that c **a** d and c **b** d appeared in the training set (as identified in the substitution graph).
- There should not be any substring of **a** or **b** in the set of constituents except when **a** is a substring of **b**.

These techniques enabled Clark to identify the target grammars exactly, despite the lack of any explicit negative examples.

As a result of his involvement in the Omphalos competition and the success of his techniques, Clark believes that with further research he will be able to prove that strict deterministic languages are PAC-learnable when the distributions of the training examples are generated by suitable probabilistic deterministic push down automata.

Additionally he envisions that his techniques could be extended into more powerful algorithms with (string) sample complexity that could work left-to-right and have provable convergence properties.

6. Conclusions

The main conclusions that can be drawn from the competition is that it has achieved all of the goals that were attempted. The competition has

[7] in the identification in the limit scenario.

provided a forum for the comparison of different GI algorithms on a given task. It has brought some existing techniques to the foreground [35] and has resulted in the refinement of these and other techniques. It has also stimulated research into the field in general. For instance the number of competitors who downloaded the competitor data suggests that many non-GI researchers attempted to solve the problems. In particular four of the eight problems solved during the lifetime of the competition were solved by a competitor (Tjong Kim Sang) who has not published any prior research on grammatical inference. The competition has also defined a method of calculating the complexity of grammatical inference tasks, and because of the participation rate we can conclude that the complexity of the highest ranked competition problem is indicative of the complexity of GI problems that can be solved using state-of-the-art techniques.

As with all small scale experiments when we consider the statistical significance of the results, we know that we can't confidently state these conclusions with certainty. For instance being able to identify two target grammars exactly does not imply that all deterministic context-free languages can be PAC learnt using these techniques. Similarly the fact that some algorithms performed better on these tasks than others does not imply that this will be the case on all target languages, or even the majority of other languages. However we believe that due to the participation rates, and the repeated success of the winning technique, that the competition gives valuable insights into the state-of-the-art of the field.

6.1. Lessons Learnt

One important lesson learnt from the competition with respect to the setting of grammatical inference competitions is that great care needs to be taken in the generation of negative examples for training and particularly for testing purposes. This probably also applies to other classification and modelling tasks, where the majority of randomly selected examples are negative examples.

Both Sánchez and Tjong Kim Sang solved the easier problems using techniques that had approximated the target language with a regular language. It was therefore known that although they had correctly classified the test examples, they did not have an accurate model of the target languages. As organizers of the competition it was a little disappointing that the competitors did not use context-free GI techniques to solve the problems, as this does not progress that state-of-the-art in context-free grammatical inference. This was however a possibility that we were prepared for. We had deliberately designed the competition so that not only would differing GI techniques be pitted against one another, but also GI techniques would be pitted against non-GI techniques, such as generic classification techniques and even solving the problems by hand.

After problems 1,3,4,5 and 6 had been solved a great deal of effort was put into constructing test data sets that were more accurate indications of whether or not the competitor had successfully constructed an accurate model of the target language. Specifically new problems were added to the competition on April 14^{th} and on June 7^{th} additional test sets for problems 1 to 6 were added to the competition. These test sets became problems 6.1 to 6.6. At the completion of the competition it was discovered that when Clark had solved problems 6.4 and 6.6 he had accurately identified the target languages exactly. Similarly the techniques used by Sánchez and Tjong Kim Sang could not be used to solve these harder problems. This suggests that the strategy of careful selection of test data, in particular negative data, was successful.

6.2. Recommendations for future competitions

Over the last decade, much of the research in machine learning has focused on problems like classification and regression, where the prediction is a single univariate variable. A challenge for machine learning in the next decade is to transfer successes in these areas into the area of inferring complex objects like grammars, trees, sequences, or orderings. Perhaps this is one reason why grammatical inference is becoming a more active area of research. In this respect regular grammars are ideal for modelling sequences. Context-free grammars are also well suited to modelling sequences, but in addition can be used to model hierarchical structures within sequence (such as the phrase structure of language). Future grammatical inference competitions like the Omphalos competition can continue to play a part in the development of the field, by providing common tasks for re-

searchers to focus on, thus providing a good means of comparing techniques.

At the end of the Omphalos session at ICGI 2004 a discussion was held to provide the opportunity for those present at the conference to provide feedback to the Omphalos committee members. Topics of discussion included "was the competition a success", "how could the competition be improved for subsequent years" and "should the competition be continued for subsequent ICGI conferences". There was a general agreement that the shared task was a good opportunity to compare algorithms and promote the development of new techniques and that the competition should be continued. As a result we will couch the received comments in terms of how subsequent GI competitions were held, in the remainder of this section.

There was a request for the competition tasks to have a practical application focus, and that the application should dictate the metric used to evaluate the task. In this case the practical application would dictate the form in which training and testing data should take. For instance if the task is to infer a grammar for use with a speech recognition system, then speech recognition accuracy would be a suitable metric. Similarly if the application task is to infer the meaning of sentences, then the phrase structure of sentences is more useful than classifying whether or not a sentence is grammatical. This opinion was expressed more than one member present. This is therefore believed to be the majority opinion.

There was also a range of specific suggestions and comments made by varying individuals present. These suggestions were not universally accepted, and some suggestions made by some people were in conflict with other suggestions made by other people. These specific suggestions were as follows.

- There was a request for more than one type of data, e.g. partially structured data, in the belief that with a greater variety of data more people will compete.
- Some people expressed the opinion that if more than one type of data were available then the number of competitors per task would be low.
- There was a request for the task to be an approximation (rather than an identification) task.
- There was a request for the winner of the task to be the competitor that obtained the best accuracy.
- There was a request for the task to be related to identifying the structure of language. That is identification of a grammar rather than of a language. In this case the target language could be regular or finite acyclic languages.
- There was a complaint that the alphabet size was too small to be comparable to real world tasks.
- There was a request for the task to be one in which the training and test sets contained noise.

As a result of these discussions it is planned to continue the GI competition tradition and include another GI competition at the next ICGI to be held in 2006. At this point it is planned to, as discussed at ICGI, to model the competition tasks on real world applications. There will most likely be two competition tasks, one in the area of natural language, and the other in computational biology (DNA or protein sequences). Ideally we would like to use real world data in the competition, but this approach suffers from several problems. The first is that competitors may be able to bring external knowledge to the task at hand in way that does not encourage the development of the state-of-the-art. For instance native English speakers would have no problem labelling grammatical and ungrammatical English sentences. Another major problem is that "real" data, like DNA sequences and natural language have an inherent bias that is unknown beforehand. Also it becomes difficult to scale the competition problems to a size that is just outside the state-of-the-art, when there is no control over the underlying target model.

Acknowledgments

The contribution of Alexander Clark in reviewing the section on his winning technique is hereby acknowledged.

References

[1] E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[2] Yasubumi Sakakibara. Recent advances of grammatical inference. *Theoretical Computer Science*, 185(1):15–45, 1997.

[3] C. de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 2005. Special Issue on Grammatical Inference Techniques and Applications.

[4] D. Angluin. Queries revisited. *Theoretical Computer Science*, 313(2):175–194, 2004.

[5] L. G. Valiant. A theory of the learnable. *Communications of the Association for Computing Machinery*, 27(11):1134–1142, 1984.

[6] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[7] L. Pitt and M. Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. In Johnson [39], pages 421–432.

[8] L. Pitt and M. Warmuth. Reductions among prediction problems: On the difficulty of predicting automata. In *Third Conference on Structure in Complexity Theory*, pages 60–69, 1988.

[9] M. Kearns and L.G. Valiant. Cryptographic limitation on learning boolean formulae and finite automata. In Johnson [39], pages 433–444.

[10] R. Parekh and V. Honavar. Learning DFA from simple examples. In *Workshop on Automata Induction, Grammatical Inference, and Language Acquisition, ICML-97*, 1997.

[11] C de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27:125–138, 1997.

[12] C. de la Higuera and J. Oncina. Inferring deterministic linear languages. In Jyrki Kivinen and Robert H. Sloan, editors, *Fifteenth Annual Conference on Computational Learning Theory (COLT 2002)*, pages 185–200. 2002.

[13] B. Starkie. Left aligned grammars—identifying a class of context-free grammar in the limit from positive data. In C. de la Higuera, P. W. Adriaans, M. van Zaanen, and J. Oncina, editors, *Proceedings of the Workshop and Tutorial on Learning Context-Free Grammars held at the 14th European Conference on Machine Learning (ECML) and the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD); Dubrovnik, Croatia*, pages 89–101, 2003.

[14] B. Starkie and Henning Fernau. The boisdale algorithm—an induction method for a subclass of unification grammar from positive data. In G. Paliouras and Y. Sakakibara, editors, *Grammatical Inference: Algorithms and Applications; Seventh International Colloquium, ICGI 2004*, volume 3264, pages 235–247. Berlin, Germany: Springer, Athens, Greece, 2004.

[15] J. M. Sempere and P. García. Learning locally testable even linear languages from positive data. In Adriaans et al. [38], pages 225–237.

[16] T Yokomori. Polynomial-time identification of very simple grammars from positive data. *Theoretical Computer Science*, 298:179–206, 2003.

[17] P. W. Adriaans. *Language Learning from a Categorial Perspective*. PhD thesis, University of Amsterdam, Amsterdam, the Netherlands, November 1992.

[18] A. Clark. *Unsupervised Language Acquisition: Theory and Practice*. PhD thesis, University of Sussex, Brighton, UK, 2001.

[19] D. Klein and C. D. Manning. Distributional phrase structure induction. In CoNLL [37], pages 113–120.

[20] K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm. In V. Honavar and G. Slutzki, editors, *Proceedings of the Fourth International Conference on Grammar Inference*, volume 1433 of *Lecture Notes in AI*, pages 1–12, Berlin Heidelberg, Germany, 1998. Springer-Verlag.

[21] J. J. Horning. *A study of grammatical inference*. PhD thesis, Stanford University, Stanford:CA, USA, 1969.

[22] M. van Zaanen, A. Roberts, and E. Atwell. A multilingual parallel parsed corpus as gold standard for grammatical inference evaluation. In Lambros Kranias, Nicoletta Calzolari, Gregor Thurmair, Yorick Wilks, Eduard Hovy, Gudrun Magnusdottir, Anna Samiotou, and Khalid Choukri, editors, *Proceedings of the Workshop: The Amazing Utility of Parallel and Comparable Corpora; Lisbon, Portugal*, pages 58–61, May 2004.

[23] N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2:113–124, 1956.

[24] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Publishing Company, Reading:MA, USA, 2001.

[25] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York:NY, USA, 1997.

[26] A. Stolcke. Boogie. ftp://ftp.icsi.berkeley.edu/pub/ai/stolcke/software/boogie.shar.Z, 2003.

[27] K. Nakamura and M. Matsumoto. Incremental learning of context-free grammars. In Adriaans et al. [38], pages 174–184.

[28] C. M. Cook, A. Rosenfeld, and A. R. Aronson. Grammatical inference by hill climbing. *Informational Sciences*, 10:59–80, 1976.

[29] C. Donnelly and R. M. Stallman. *Bison Manual: Using the YACC-compatible Parser Generator, for Version 1.875*. GNU Press, Boston, MA, USA, 2003.

[30] A. Denise, F. Sarron, and Y. Ponty. GenRGenS. http://www.lri.fr/d̈enise/GenRGenS/, 2001. Laboratoire de Recherche en Informatique.

[31] M. J. Nederhof. Practical experiments with regular approximation of context-free languages. *Computational Linguistics*, 26(1):17–44, 2000.

[32] F. Otto. On deciding the confluence of a finite string-rewriting system on a given congruence class. *Journal of Computer and System Sciences*, 35:285–310, 1987.

[33] A. Clark. Learning deterministic context free grammars: The omphalos competition. *submitted to Machine learning*, 2005.

[34] R. Eyraud, C. de La Higuera, and J. Janodet. Representing languages by learnable rewrite systems. In G. Paliouras and Y. Sakakibara, editors, *Grammatical Inference: Algorithms and Applications; 7th International Colloquium, ICGI 2004*, volume 3264, pages 139–150, Athens, Greece, 2004. Berlin, Germany: Springer.

[35] A. Clark. Unsupervised induction of stochastic context-free grammars using distributional clustering. In CoNLL [37], pages 105–112.

[36] M. van Zaanen. ABL: Alignment-Based Learning. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING); Saarbrücken, Germany*, pages 961–967. Association for Computational Linguistics (ACL), July 31–August 4 2000.

[37] Association for Computational Linguistics (ACL). *Proceedings of the Workshop on Computational Natural Language Learning held at the 39th Annual Meeting of the Association for Computational Linguistics (ACL) and the 10th Meeting of the European Chapter of the Association for Computational Linguistics (EACL); Toulouse, France*, July 2001.

[38] P. W. Adriaans, H. Fernau, and M. van Zaanen, editors. *Grammatical Inference: Algorithms and Applications (ICGI); Amsterdam, the Netherlands*, volume 2482 of *Lecture Notes in AI*, Berlin Heidelberg, Germany, September 23–25 2002. Springer-Verlag.

[39] D. S. Johnson, editor. *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, New York, NY, USA, 1989. ACM Press.