Ordered Semantic Hyper Linking

David A. Plaisted and Yunshan Zhu
Computer Science Department
University of North Carolina
Chapel Hill, NC 27599-3175
{plaisted,zhu}@cs.unc.edu
Fax: (919)962-1799

Abstract

In this paper, we present a novel first order theorem proving strategy – ordered semantic hyper linking. Ordered semantic hyper linking (OSHL) is an instance-based refutational theorem proving strategy. It is sound and complete. OSHL has an efficient propositional decision procedure. It solves first order problems by reducing them to propositional problems. It uses natural semantics of an input problem to guide its search. It also incorporates term rewriting to handle equality. The propositional efficiency, semantic guidance and equality support allow OSHL to solve problems that are difficult for many other strategies. The efficiency of OSHL is supported by experimental study as well as complexity analysis.

Introduction

In this paper, we present a novel first order theorem proving strategy – ordered semantic hyper linking. Ordered semantic hyper linking (OSHL) is an instance-based refutational theorem proving strategy. It is sound and complete. OSHL is propositionally efficient. It uses natural semantics of input problems to guide its search. It also incorporates term rewriting to handle equality.

Many current first order theorem proving strategies are inference- based strategies, that is, reasoning is carried out directly at the first order level. Ordered semantic hyper linking is an instance-based strategy. It generates instances of first order clauses, and thus reduces first order problems to propositional problems. Ordered semantic hyper linking is an efficient decision procedure for propositional problems. It proves unsatisfiability by exhaustively contradicting all possible models.

Ordered semantic hyper linking imposes orderings on ground literals and interpretations. The ordering information is used to control the model finding process. We show that models generated by OSHL are monotonically increasing. Ordering also facilitates the use of term rewriting in handling equality theorems. A user can optionally provide an input semantics for a problem. Ordered semantic hyper linking uses the semantics to guide its search. Any interpretation of the input clauses can be used as an input semantics. An interpretation is represented as a decision procedure of ground literals. A natural semantics often greatly reduces the search space of OSHL.

There has been some interest recently in studying propositional decision procedures. Although propositional satisfiability test is an NP-complete problem, (Selman & Kautz 1993) has showed that many propositional problems can be solved quickly. Several efficient propositional provers have been implemented(Zhang & Stickel 1994) (Selman & Kautz 1993), and they perform very well on a wide range of problems. The prover in (Zhang & Stickel 1994) is based on the Davis-Putnam method, and the prover in (Selman & Kautz 1993) is based on randomized local search. There are some also recent work in applying efficient propositional procedure in various domains. In (Kautz, McAllester, & Selman 1996), planning problems are converted into propositional problems, and efficient propositional decision procedures are used to tackle those problems. Interestingly, that approach shares the same principle as OSHL and instance-based strategies in general. In OSHL, the original problems are expressed in first order logic, the conversion to propositional problems is automatic, and the domain knowledge is represented in the form of input semantics. OSHL might not be as efficient as the method of (Kautz, McAllester, & Selman 1996) on specific problems, but is perhaps more general.

The history of instance-based theorem proving strategies traces back to (Gilmore 1960). Most early attempts were based on enumerating Herbrand instances, and they have not been very successful partially due to the lack of unification and partially due to the lack of control in instance generation. (Wang & Bledsoe 1987; Slagle 1967; Chu & Plaisted 1994) used semantics to guide the search of theorem provers. The semantics allowed in these provers were somewhat limited. CLIN (Lee & Plaisted 1992) uses a strategy based on clause linking. CLIN-S (Chu & Plaisted 1994) adds semantic support to clause linking. RRTP (Paramasivam & Plaisted 1997) generates instances using replacement rules and uses the propositional decision procedure in (Zhang & Stickel 1994) to detect propositional unsatisfiability. Ordered semantic hyper linking improves upon CLIN and CLIN-S. CLIN-S restricts input semantics to be either finite or decidable. OSHL allows any semantics that can be expressed as a ground decision procedure. OSHL imposes an ordering on all ground literals. The ordering allows OSHL to have a more systematic model generation strategy. OSHL also permits the use of term rewriting to handle equalities.

We compare OSHL with OTTER(McCune 1990), which

is a state of the art resolution-based theorem prover. OTTER is very efficient in handling Horn problems and UR resolvable problems. It is not as efficient for non-Horn problems. This is particularly evident on examples like the pigeon-hole problems. Probably the best way to solve such problems is to do a systematic case analysis. However, resolution does inference by combining literals from different clauses and generating resolvents. The equivalence of a single case analysis is essentially duplicated many times in the generated resolvents. OSHL uses an efficient propositional decision procedure. It is particularly good at near-propositional problems, whether it is Horn or non-Horn. OSHL reduces first order problems to propositional problems by generating instances. The smallest instances are generated first. For problems that require large instances, too many small instances could be generated. Natural semantics and term rewriting often greatly reduce the number of instances generated. For example, with an input semantics that models the input axioms, OSHL only generates goal-related instances.

The paper is organized as follows. In the next section, we provide background and define orderings used in our implementation. We introduce ordered semantic hyper linking as a propositional decision procedure and discuss the use of semantics in the guiding the search. We then extend the description to first order problems. We provide some experimental results in the end.

Background

In this section, we define some concepts related to our future discussions. We also define orderings used in our implementation of the ordered semantic hyper linking strategy. We will assume a general knowledge of first order logic and refutational theorem proving. We refer readers to (Chang & Lee 1973) for a general introduction to these areas.

A *term* is a well-formed expression containing function symbols, constant symbols, and variables, as, $f(X, g(a))$. An *atom* is a predicate symbol followed by a list of terms, as, $p(X, g(a, b))$. A *literal* is an atom or an atom preceded by a negation sign, as, $\neg q(a, X)$. A literal preceded by a negation sign is *negative* and a literal without a negation sign is *positive*. The literals L and $\neg L$ are said to be *complementary*. A *clause* is a disjunction of literals, written as a set, as, $\{\neg p(x), q(f(x))\}$. Free variables in a clause are assumed to be universally quantified. A literal(term, clause) is *ground* if it does not contain variables, as $\neg q(a, b)$.

An *interpretation* of a first order order formula consists of a domain and a set of mappings. Constants, functions and predicates symbols are mapped to elements, functions and relations defined in the domain respectively. Every interpretation has an equivalent Herbrand interpretation. A Herbrand interpretation maps a ground term to itself and a ground literal to either true or false. We use the set of true ground literals to represent the interpretation. We also use the term *semantics* for interpretation. A *trivial semantics* is an interpretation that assigns true to all positive literals or to all negative literals. An interpretation I models a formula C if C is true under the interpretation. We also say I is a model of C, or $I \models C$. We use $I(s)$ to represent the element denoted by term s. We use I_0 to name the input semantics. A current semantics is represented as $I_0[EL]$, where EL stands for *eligible literals*. Some of these notations were also used in (Chu & Plaisted 1994).

Definition 1.1 *Given an interpretation I of $\mathcal{A}$ and literals L_i over $\mathcal{A}$, let $I[L_1, L_2, \ldots, L_n]$ be defined as follows : $I[L_1, L_2, \ldots, L_n] \models L$ iff*
(a) $L \in \{L_1, \ldots, L_n\}$ or
(b) $L \notin \{L_1, \ldots, L_n\}$, $\neg L \notin \{L_1, L_2, \ldots, L_n\}$ and $I \models L$.
thus $I[L_1, \ldots, L_n]$ is like I except for the finite list $[L_1, L_2, \ldots, L_n]$ of "exceptions".

Orderings are defined over ground terms, literals, clauses and interpretations. Different orderings can be used with ordered semantic hyper linking. We here define the orderings used in our implementation.

Definition 1.2 *The linear size of a term(atom) t, $size(t)$, is the length of the term(atom) written as a character string, ignoring commas and parentheses.*

Definition 1.3 *A size-lexicographic ordering $<$ on ground terms(atoms) is defined as follows: $s < t$ iff (a) $size(s) < size(t)$ or (b) $size(s) = size(t)$ and $s <_{lex} t$, where $<_{lex}$ is a lexicographic ordering. Ordering $<$ is extended to literals as follows: $\neg s < t$ iff $s < t$; $s < \neg t$ iff $s < t$; $s < \neg s$, where s and t are atoms*

Definition 1.4 *Assume interpretation $I = I_0[L_{i1}, L_{i2}, \ldots, L_{in}]$, $J = I_0[L_{j1}, L_{j2}, \ldots, L_{jm}]$. L_is and L_js are sorted by the size-lexicographic ordering. $I < J$ iff*
(a) $n = 0$ $(I = I_0)$ and $J \neq I_0$ or
(b) $L_{i1} > L_{j1}$ or
(c) $L_{i1} = L_{j1}$ and $I_0[L_{i2}, \ldots, L_{in}] < I_0[L_{j2}, \ldots, L_{jm}]$.

The ordering on non-tautologous ground clauses is defined as a multiset extension of the ordering defined on literals. We now give a few examples. Literals: $p(a) < q(a) < \neg q(a) < r(a, a) < \neg r(a, a)$; Clauses: $\{q(a), p(a)\} < \{\neg q(a), p(a)\} < \{r(a, a), q(a)\}$; Interpretations: $I_0[p(a), q(a)] > I_0[q(a), r(a, a)]$.

In the following sections, we describe ordered semantic hyper linking. We will first explain the algorithm as a propositional procedure, and then extend it to handle first order problems. This seems to be a good way to introduce various components of the algorithm, and it also reflects the design philosophy of ordered semantic hyper linking. We also use a section to study the effect of semantics in guiding the search of OSHL.

Propositional Procedure

Figure 1 shows the Ordered Semantic Hyper Linking procedure. In this section, we assume the input clauses are all propositional.The proof procedure is essentially a model finding procedure. Starting with an input semantics, the proof procedure iteratively finds a new clause that contradicts the current semantics and updates the semantics to satisfy the new clause. We call the first part of the iteration *instance generation* and the second part *model generation*. The iteration continues until no more contradicting clauses exist or no more models exist. In the former case, the clause set is satisfiable; and in the latter case, it is unsatisfiable.

For propositional problems, instance generation (procedure *mi*) is straightforward. It only needs to pick an input clause that contradicts the current semantics. A clause C contradicts the current semantics I iff every literal L in C is false in I. Formally, $I \not\models C$ iff $\forall L \in C, I \not\models L$. A Current semantics I is represented as $I_0[EL]$, where I_0 is the input semantics and EL is the set of eligible literals. For

```
% S is the set of input clauses
% I0 is the initial semantics
I = I0;      % I is the current semantics
G = ∅;       % G is the set of ground instances
while ({} ∉ G)
      {
      % instances generation
      D = mi(S, I);  % D = sθ, s ∈ S, and I ⊭ D
      if D = ∅ then return satisfiable fi;
      % model generation
      G = simp(G, D);
      I = I0[lm(G)];  % I is the least model of G
      }
return unsatisfiable.
```

Figure 1: Ordered Semantic Hyper Linking procedure

```
procedure simp([C1, C2, ..., Cn], D)
    if max(Cn) and max(D) are complementary then
        D ← a_res(Cn, D);
        return simp(C1, C2, ..., Cn − 1], D)
    else if max(Cn) > max(D) then
            return simp([C1, C2, ..., Cn−1], D)
            else return[C1, C2, ..., D]
        fi
    fi
end simp
```

Figure 2: Model generation

propositional problems, both I_0 and EL are finite sets. To check if a literal L contradicts the current semantics I is trivial: $I \not\models L$ iff ($L \notin EL$ and $L \notin I_0$) or ($\neg L \in EL$). Instance generation can be done fairly efficiently. A linear search through all input clauses takes quadratic time. Slightly more intelligent searches involving marking the literals take only linear time. Of course, this cost occurs every iteration.

Definition 2.1 *Suppose C and D are ground clauses and suppose there is a literal L such that $L = max(C)$ and $\neg L = max(D)$, where $max()$ represents the maximal literal in a clause. Then $a\_res(C, D) = (C - \{L\}) \cup (D - \{\neg L\})$*

Figure 2 shows procedure *simp*, which is the core of model generation. procedure $lm(G)$ simply returns the set of the maximal literals of each clause in G. It can be showed that $I_0[lm(G)]$ is always the minimal model(according to the ordering defined in the previous section), if a model exists, of the clauses in G.

Theorem 2.2 *As a propositional procedure, ordered semantics hyper linking is sound and complete.*

The soundness of ordered semantics hyper linking is obvious. It follows from the soundness of binary resolution. It can be showed that the current semantics is monotonically increasing. However, there is an upper bound for all interpretations. Thus the loop in Figure 1 always terminates and the algorithm is complete.

Axioms :
$S_1 : \{r, q, p\}$ $S_5 : \{r, q, \neg p\}$
$S_2 : \{\neg r, q, p\}$ $S_6 : \{\neg r, q, \neg p\}$

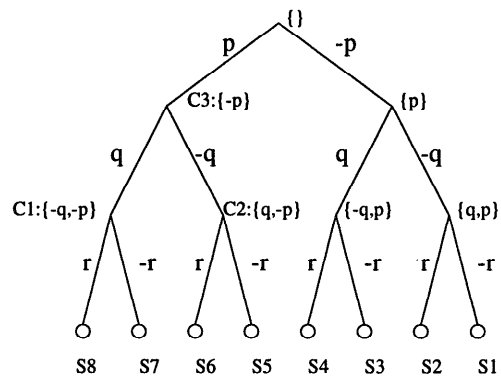

Figure 4: A semantic tree constructed by OSHL

$S_3 : \{r, \neg q, p\}$ $S_7 : \{r, \neg q, \neg p\}$
$S_4 : \{\neg r, \neg q, p\}$
Theorem :
$S_8 : \{\neg r, \neg q, \neg p\}$

Example 4: A propositional problem

We now use a simple example to illustrate the ordered semantic hyper linking algorithm. The set of clauses for our example is showed as above. We choose the input semantics I_0 to be $\{p, q, r\}$. We regard the first seven clauses as axioms and the last clause as the theorem, and I_0 models all the axioms and contradicts the theorem. According to the ordering defined in the background section, $p < q < r$. Literals in the input clauses are ordered. We call each iteration of the loop in Figure 1 a round. We use subscripts to distinguish each round's variables.
Round 1: $I_1 = I_0$, $D_1 = \{\neg r, \neg q, \neg p\}$, $I_{new} = I_0[\neg r]$
Round 2: $I_2 = I_0[\neg r], D_2 = \{r, \neg q, \neg p\}, I_{new} = I_0[\neg q]$. In the model generation procedure, D_2 is resolved with D_1, and a new clause $C_1\{\neg q, \neg p\}$ is generated.
Round 3: $I_3 = I_0[\neg q], D_3 = \{\neg r, q, \neg p\}, I_{new} = I_0[\neg q, \neg r]$. No resolution is done in this round, D_3 is simply added to the set of ground clauses G. Note that A-resolution only resolves the maximal literals of two clauses.
Round 4: $I_4 = I_0[\neg q, \neg r], D_4 = \{r, q, \neg p\}, I_{new} = I_0[\neg p]$. In the model generation procedure, D_4 is resolved with D_3 to generate clause C_2 $\{q, \neg p\}$, C_2 subsequently is resolved with the C_1 generated in round 2. A new clause $\{\neg p\}$ is generated.
In the next 4 rounds, an empty clause will be generated, and no more models can be found. Thus refutation is obtained.

Figure 4 shows the semantics tree that corresponds to the search of the OSHL procedure. Each leaf node of the semantic tree corresponds to an input clause, and each internal node corresponds to an A-resolvent. In fact, the proof process by ordered semantic hyper linking can be understood as a process of constructing and searching through a semantic tree. An interpretation corresponds to a branch of the semantic tree. Assuming that the input semantics is the left-most branch of the semantic tree and the nodes are in an increasing order from the root to the leaves, ordered semantic hyper linking performs a depth first [1] search of

[1] In the case of first-order logic, it performs an iterative-

the semantic tree from left to right.

Although ordered semantic hyper linking uses A-resolution as a low-level operation, it is fundamentally different from many of the resolution based methods. With a fixed input semantics, a unique semantics tree for the input problem is constructed, and the search of a semantic tree is carried out in a systematic fashion. Most resolution-based strategies can be described as non-deterministic procedures. They construct semantic trees in a bottom-up fashion by non-deterministically generating resolvents. In some sense, most resolution based strategies are simultaneously constructing many semantics trees in searching for a proof. The redundancy is particularly evident when many case analysis are needed to obtain a proof. For the problem in Example 4, binary resolution will generate much more resolvents than OSHL. Hyper-resolution greatly reduces the number of possible resolvents, but it still generates redundant clauses, and subsumption tests are needed to remove the redundant clauses. (Plaisted 1994) has studied the duplication in various propositional strategies.

The ground procedure of ordered semantic hyper linking is in essence similar to the Davis-Putnam method (Davis & Putnam 1960). Both are modeling finding strategies, and prove unsatisfiability by exhausting all possible models. Both use case analysis to handle non-Horn clauses, although in somewhat different formats. By ordering the terms, ordered semantic hyper linking has a more systematic search than Davis-Putnam method. Ordered semantic hyper linking can also utilize semantical guidance in its search process.

Semantics

Ordered semantic hyper linking can use semantics to guide its search. In this section, we will show what a "good semantics" is and how it helps to guide the search process.

Let's again use a simple example as an illustration.

Axioms:
$\{\neg a, a_1\}$. $\{\neg b, b_1\}$. $\{\neg c, c_1\}$.
$\{\neg a_1, a_2\}$. $\{\neg b_1, b_2\}$. $\{\neg c_1, c_2\}$.
$\{\neg a_2, d\}$. $\{b\}$. $\{\neg c_2, d\}$.
$\{a\}$.
Theorem : $\{\neg d\}$

A forward chaining technique will derive $\{a, b, a_i, b_i, d\}$. b_is are logical consequences of the input axioms, but they are not related to the theorem. A backward chaining technique will try subgoals $\{\neg d, \neg a_i, \neg c_i, \neg a\}$. c_is are relevant to the theorem, however they are not logical consequences of the input axioms.

We outline the search of ordered semantic hyper linking using $\{a, a_i, b_i, d\}$ as the input semantics.
Round 1. $I = I_0$, $D_1 = \{\neg d\}$, $I_{new} = I_0[\neg d]$
Round 2, $I = I_0[\neg d]$, $D_2 = \{\neg a_2, d\}$, $I_{new} = I_0[\neg d, \neg a_2]$
Round 3, $I = I_0[\neg d, \neg a_2]$, $D_3 = \{\neg a_2, d\}$, $I_{new} = I_0[\neg d, \neg a2, \neg a1]$
Round 4, $I = I_0[\neg d, \neg a_2, \neg a_1, \neg a]$, $D_4 = \{a\}$, empty clause is derived.

With the guidance of a "good" input semantics, only subgoals $\{\neg d, \neg a_2, \neg a_1, \neg a\}$ are attempted. All of them are relevant to the theorem and refutable based on the input axioms. We now formally define several concepts.

---

deepening search.

Assume that a set of clauses S contains an axiom set A and a theorem set T. S_H is the set of all Herbrand instances of S. If S is propositional, S_H and S are identical. A connection graph CG contains clauses in S_H as nodes. There is an edge between two nodes if they contain complementary literals. A ground clause C is *goal-related* if C is an instance of a clause in T, or C is connected to a goal-related clause in CG, or C is a resolvent of two goal-related clauses. We say the search of OSHL is *goal sensitive* if it only generates goal-related clauses, that is, the clause set G in Figure 1 contains only goal-related clauses. A clause C is *proof-related* if $\forall L \in C, A \models \neg L$ (or $A \models \neg C$). We say the search of OSHL is *proof sensitive* if it only generates proof-related clauses. If A is a Horn set, the minimal model is defined as the intersection all models of A. For a Horn set, the minimal model of A is indeed a model for A.

Theorem 3.1 *For a set of clauses S with an axiom set A, OSHL is goal sensitive if the input semantics I_0 is a model for A.*

Theorem 3.2 *For a set of Horn clauses S with an axiom set A, OSHL is goal sensitive and proof sensitive if the input semantics I_0 is a minimal model for A.*

Theorem 2.2 in the last section showed that OSHL is complete no matter what input semantics is used. Theorem 3.1 and Theorem 3.2 show that better input semantics makes the prover more efficient. In terms of the construction of semantic trees, a good input semantics reduces unnecessary branching in the semantic tree. If the input semantics is a model for the axioms, the tree only branches at literals of goal-related clauses. If the input semantics is a minimal model, the tree only branches at goal-related and proof-related literals. Thus with a better input semantics, OSHL can construct a smaller semantic tree and have a more efficient search.

For Horn clauses, the minimal model of the input axioms is the best input semantics, but the minimal model is usually unknown. After all, deriving a minimal model is equivalent of deriving all logical consequences of the Horn theory. However, an approximation of the minimal model is often available, and it is often good enough to greatly reduce the search space. We will give an example of natural semantics in the next section where we cover first order problems.

First Order Procedure

We described ordered semantic hyper linking as a propositional procedure in previous sections, and we now extend it to handle first order problems. Many of the inference-based strategies are lifted version of their ground counterparts and do inference at the first order level. Ordered semantic hyper linking reduces first order problems to propositional problems and does inference at the propositional level.

Ordered semantic hyper linking contains two main components: instance generation and model generation. The overall structure of the first order procedure is identical to that of the propositional procedure showed in Figure 1. The only differences are in instance generation. In the propositional procedure, all input clauses are ground. Instance generation only involves picking the right input clause. In the first order procedure, unification and instantiation are needed to generate contradicting instances from the first order input clauses. We show the procedure of instance generation in Figure 5.

```
procedure mi(S,I)
I = I0[EL];
% max is the maximal size of the instance
for max size = 1,2,3, ... do
   for all C in S do
    for all partitions (C1', C2) of C do
      unify literals of C1 with complements of EL
      for all C1α thus obtained do
       instantiate C2α such that I0 ⊭ D and
             D ∩ EL = ∅
       if succeeds then return(C1α ∪ D)
end mi
```

Figure 5: first order instance generation procedure

Theorem 4.1 *A ground clause D contradicts the current semantics $I_0[EL]$, iff $D = D_1 \cup D_2$, $D_1 \cap D_2 = \emptyset$, $\forall L \in D_1, \neg L \in EL$, and $I_0 \not\models D_2$.*

As showed in the above Figure 5, the instance generation module loops through all input clauses to find a instance that contradicts the current semantics. The max size bound guarantees that an instance with the smallest size is generated first. To generate a contradicting instance of a non-ground clause C. C is split into C_1 and C_2. If the input semantics models C, C_1 has to contain at least one literal. The literals in C_1 are unified with the negation of eligible literals, and a substitution α is obtained. $C_1\alpha$ is ground. $C_2\alpha$ is further instantiated to $D = C_2\alpha\beta$ such that $I_0 \not\models D$ and $D \cap EL = \emptyset$. In general, β is found by enumeration. The enumeration is done in two stages. First variables in $C_2\alpha$ are replaced with semantically minimal terms; and then for the instances that contradict the input semantics, the minimal terms are replaced with non-minimal terms of the same interpretation. A term s is semantically minimal if s is the smallest term with the interpretation $I(s)$. If C_1 contains all the variables of C, $C_2\alpha$ is ground and no enumeration is needed. This case often occurs when C represents the definition of a concept and C_1 contains the concept being defined. If the input semantics is decidable, the enumeration can also be avoided.

Theorem 4.2 *OSHL is sound and complete.*

As in the propositional procedure, the instance generation process is guided by the input semantics. We've showed that how semantics helps in guiding the search of the propositional procedure. The same results hold for the first order case.

Remark 4.3 *Theorem 3.1 and Theorem 3.2 are still valid when S contains first order clauses.*

We now give an example in first order logic.

Axioms:
$\{p(e, X, X)\}$.
$\{p(i(X), X, e)\}$.
$\{p(U, Z, W), \neg p(X, Y, U), \neg p(Y, Z, V), \neg p(X, V, W)\}$.
$\{p(X, V, W), \neg p(X, Y, U), \neg p(Y, Z, V), \neg p(U, Z, W)\}$.
$\{p(X, Y, f(X, Y))\}$.
$\{p(a, b, c)\}$.[2]
Theorem:
$\{\neg p(a, e, a)\}$.

[2]The last two axioms are not needed for the proof.

The above example represents the following group theory problem : a group with left inverse and left identity has a right identity. A natural semantics for the problem can be constructed based on an example from the integer addition group. The domain is the set of integers. e denotes 0, a denotes 2, b denotes 3, c denotes 5, function $i()$ denotes the inverse function, function $f()$ denotes the addition function, and predicate p denotes the sum relation. The semantics is represented as decision procedures for ground literals. A Prolog representation of the procedures is showed below. Integer addition group is commutative. $p(a, b, f(b, a))$ is true in the given semantics, but it is not true in the minimal model. A better semantics, i.e. a better approximation of the minimal model, can be constructed based on free group examples.

$eval(e, 0)$.
$eval(a, 2)$.
$eval(b, 3)$.
$eval(c, 5)$.
$eval(i(Xt), V)$:- $eval(Xt, X)$, V is $-X$.
$eval(f(Xt, Yt), V)$:- $eval(Xt, X), eval(Yt, Y)$,
V is $X + Y$.
$eval(p(Xt, Yt, Zt))$:- $eval(Xt, X), eval(Yt, Y)$,
$eval(Zt, Z)$, $Temp$ is $X + Y$,
$Z = Temp$.

Discussion and Results

We now comment on the complexity of ordered semantic hyper linking. First order logic has only partial decision procedures. The run time of OSHL can not be bounded recursively by the size of the input clauses. We define *maximal literal size* of a set of clauses S to be the maximum of the size of the literals in S. We define the $c(S)$ to be the minimum, over all Herbrand sets T for S, of the maximal literal size of T. By Herbrand's theorem, $c(S)$ is finite if S is unsatisfiable. We can analyze the complexity of OSHL by including, in the input string, $c(S)$ represented in unary along with the input clause set S. We call it the *complexity with respect to linear term size measure*(Plaisted & Zhu 1997). OSHL is double exponential with respect to the linear term size measure. There are at most $O(2^{c(S)})$ ground instances of S of term size $c(S)$ or less, thus the instance generation module takes at most $O(2^{c(S)})$ time and the model generation module takes at most $O(2^{2^{c(S)}})$ time. It is co-NEXPTIME hard to determine whether a set S has a Herbrand set with a maximal literal size of less than n. Thus the double exponential complexity of OSHL is probably the best we can get. In practice, due to the use of an efficient propositional procedure, the model generation module usually takes much less time than the instance generation module. In fact, we might assume an efficient proposition procedure to have an expected polynomial complexity (especially when there are many Horn clauses), thus OSHL will have an expected single exponential complexity.

Ordered semantic hyper linking has been implemented in Prolog. The input is the set of clauses and optionally a semantics provided by the user. The input semantics is represented as a ground decision procedure as showed in the last section. If the prover finds a refutation, it prints out the proof. The main modules are instance generation and model generation. We have added equality support to the prover. It includes term rewriting, critical pair generation and narrowing. Term rewriting reduces the set of

Problem	# of instances	OSHL Time	OTTER Time
EXQ1(SYN013-1)	95	35.0	3000+
EXQ2(SYN014-1)	241	305.9	3000+
EXQ3(SYN015-1)	280	383.9	3000+
SET159-6	101	95.5	3000+
SET169-6	67	160.4	3000+
SET249-6	93	97.1	3000+
wos19(GRP039-3)	5	38.2	3000+
wos20(GRP040-4)	61	882.3	950.7
IMV(ANA002-3)	188	317.2	3000+
e4(GRP002-4)	n/a	801.0	2.0
e5(BOO002-1)	n/a	83.6	21.2
e6(RNG009-7)	n/a	245771	3000+

Table 1: Timing of OSHL and OTTER. Time is measured in seconds on a SPARC-20. 3000+ means that no proof is found in 3000 seconds. In the second column, we list the number of instances that OSHL generates in searching for a proof.

instances generated by OSHL, and critical pair generation and narrowing maintain the completeness of the prover. The prover also contains UR resolution and replacement rules. They can be turned on or off via flag settings. Due to the limit of space, we do not discuss these procedures in this paper.

The OSHL prover has solved a number of interesting problems. Although some hard problems can be solved using a trivial semantics, the prover performs best when a natural semantics is used. We generated natural semantics for many problems by hand. The process is easier than it sounds. For example, many of the GRP problems from TPTP library (Sutcliffe, Suttner, & Yemenis 1993) have the same set of axioms. Had the library used a uniformed set of naming scheme for constant symbols (not just functions and predicates), a single semantics would have sufficed for all the GRP problems! We hope that the TPTP library will eventually include semantics for the appropriate problems in the database.

Table 1 shows some of the problems that OSHL solved. The EXQ problems and the SET problems demonstrate the propositional efficiency of the prover. The replacement module helped in solving the SET problems. It essentially generates an unsatisfiable set of propositional clauses by expanding the definitions. OSHL solves 71% of the 413 SYN problems in TPTP library. A trivial semantics is used for the run. The wos problems and the intermediate value theorem problem show the effectiveness of semantic guidance. We used free groups for the wos problem and a piece-wise function for the IMV. Natural semantics greatly reduces the number of instances generated by OSHL. e4, e5 and e6 show the effect of equality support in the prover. They, along with e1-e3, are a set of equality benchmark problems proposed by (Lusk & Overbeek 1985). None of them can be solved without the equality support. In the current implementation, OSHL is not as efficient on pure equality problems as OTTER.

Acknowledgments

This research was partially supported by NSF grant CCR-9627316.

References

Chang, C., and Lee, R. 1973. *Symbolic Logic and Mechanical Theorem Proving.* New York: Academic Press.

Chu, H., and Plaisted, D. 1994. Semantically guided first-order theorem proving using hyper-linking. In *Proceedings of the Twelfth International Conference on Automated Deduction*, 192–206. Lecture Notes in Artificial Intelligence 814.

Davis, M., and Putnam, H. 1960. A computing procedure for quantification theory. *Journal of the Association for Computing Machinery* 7:201–215.

Gilmore, P. C. 1960. A proof method for quantification theory. *IBM Journal of Research and Development* 4:28–35.

Kautz, H.; McAllester, D.; and Selman, B. 1996. Encoding plans in propositional logic. In *Proceedings of Knowledge Representation and Reasoning (KR-96).*

Lee, S.-J., and Plaisted, D. 1992. Eliminating duplication with the hyper-linking strategy. *Journal of Automated Reasoning* 9(1):25–42.

Lusk, E., and Overbeek, R. 1985. Non-horn problems. *Journal of Automated Reasoning* 1:103–114.

McCune, W. 1990. Otter 2.0 (theorem prover). In Stickel, M., ed., *Proceedings of the 10th International Conference on Automated Deduction*, 663–4.

Paramasivam, M., and Plaisted, D. A. 1997. A Replacement Rule Theorem Prover. *Journal of Automated Reasoning* Forthcoming.

Plaisted, D., and Zhu, Y. 1997. *The Propositional and First-Order Complexity of Theorem Proving Strategies.* Vieweg. forthcoming.

Plaisted, D. 1994. The search efficiency of theorem proving strategies. In *Proceedings of the Twelfth International Conference on Automated Deduction*, 57–71. Lecture Notes in Artificial Intelligence 814.

Selman, B., and Kautz, H. 1993. An empirical study of greedy local search for satisfiability testing. In *Proceedings of the Eleventh National Conference on Artificial Intelligence(AAAI-93).*

Slagle, J. R. 1967. Automatic theorem proving with renamable and semantic resolution. *Journal of the ACM* 14(4):687–697.

Sutcliffe, G.; Suttner, C.; and Yemenis, T. 1993. The TPTP problem library. Technical Report 93/11, Department of Computer Science, James Cook University, Australia.

Wang, T., and Bledsoe, W. 1987. Hierarchical deduction. *Journal of Automated Reasoning* 3:35–77.

Zhang, H., and Stickel, M. E. 1994. Implementing the Davis-Putnam algorithm by tries. Technical report, Department of Computer Science, University of Iowa.