An Empirical Study of the 3-Valued Kripke-Kleene Semantics in Random Propositional Logic Programs

extended abstract

Chris Giannella[1]

Department of Computer Science
Lindley Hall
Indiana University
Bloomington, Indiana 47405-4101
USA
E-mail: cgiannel@cs.indiana.edu
Phone Number: (812) 855-6486

John Schlipf[2]

Department of Electrical and Computer Engineering & Computer Science
814 Rhodes Hall, P.O. Box 210030
University of Cincinnati
Cincinnati, Ohio 45221-0030
USA
E-mail: John.Schlipf@UC.Edu

February 23, 1998

Abstract

We empirically studied the behavior of the 3-Valued Kripke-Kleene semantics in a parameterized distribution of random propositional logic programs. In our distribution, programs with m rules are generated from n propositional letters by repeating the following process m times: uniformly, randomly, and independently choose k letters (with replacement) from the set of n proposition letters and negate each of the last $k-1$ letters independently with probability p. The first letter becomes the head of the rule and the remaining $k-1$ literals become the subgoals. The collection of m rules make up the logic program. We fixed $p = 0.5$, $k = 3$ and varied n over a range of values investigating, for each, the relationship between m/n and the percentage of inconsistent models produced by the 3-Valued Kripke-Kleene semantics. In all of our experiments, the 3-Valued Kripke-Kleene semantics were computed with respect to a fixed percentage of proposition letters (randomly chosen) initially assigned TRUE and a fixed percentage (randomly chosen) initially assigned FALSE. We observed that there exist regions in the parameter space of our distribution in which the 3-Valued Kripke-Kleene semantics tends to produce a high percentage of inconsistent models, therefore, providing information useful to prune branches in a backtracking search for stable models. Also, we observed a region in which a very small percentage of inconsistent models are produced, therefore, providing very little information useful for pruning branches in a backtracking search.

[1]Work supported by ONR grant N00014-94-1-0382 and partially by US department of Education grant P200A502367.
[2]Work supported by ONR grant N00014-94-1-0382.

1 Introduction

For propositional logic programs, the Kripke-Kleene semantics [4] has the following property: the Kripke-Kleene partial model is a subset of the well-founded partial model [11] and, therefore, a subset of any existing stable model [6]. For this reason, the Kripke-Kleene semantics provides a simple, $O(n^2)$, approximation of the Stable Semantics where n is the number of proposition letters in the program. The Stable Semantics has received much attention of researchers in the Non-Monotonic Reasoning Community ([6], [2], [5], [7], [10] and others) since its discovery in 1988 by Gelfond and Lifschitz [6]. The interest is partly due to the fact that the Stable Semantics seems to capture some of the "common sense" reasoning with negation in rational discourse. However, the problem of finding a stable model of a general propositional logic program or determining that none exist is NP-complete [10]. The purpose of this paper is to investigate the use of the 3-Valued Kripke-Kleene semantics as a heuristic in a backtracking algorithm for finding stable models or determining that none exist.

The backtracking algorithm we consider (DP-KK) is similar to the Davis-Putnam procedure for SAT [3] and uses the 3-Valued Kripke-Kleene semantics as a heuristic for speeding the search (see Section 3 for a formal definition of the 3-Valued Kripke-Kleene semantics). Since any inconsistent partial model produced by the 3-Valued Kripke-Kleene semantics is not a subset of any existing stable model, the 3-valued Kripke Kleene semantics can be used to prune branches in the backtracking search by cutting a branch once an inconsistent partial model is found. However, a total and consistent model produced by the 3-Valued Kripke Kleene semantics is not necessarily a stable model. For this reason, any total and consistent model produced must be explicitly checked for stability (e.g. step 1 in DP-KK).

DP-KK takes an empty partial model M and a logic program L defined over a set of proposition letters P and returns "Success" if L has a stable model, and returns "Failure" if L has no stable models.

Algorithm DP-KK

1. If M is a stable model (check by the Gelfond-Lifschitz transform), Then return "Success".
2. If there is a letter, $\ell \in P$, such that $\ell, \overline{\ell} \notin M$, Then goto 3, Else return "Failure".
3. Compute the 3-Valued Kripke-Kleene partial model, M', of L with respect to $M \bigcup \{\ell\}$.
4. If M' is inconsistent (there exists a letter $p \in P$ such that: $p, \overline{p} \in M'$), Then goto 6.
5. Call DP-KK on $M \bigcup \{\ell\}$. If the call returns "Failure", Then goto 6, Else return "Success".
6. Compute the 3-Valued Kripke-Kleene partial model, M', of L with respect to $M \bigcup \{\overline{\ell}\}$.
7. If M' is inconsistent, Then return "Failure", Else call DP-KK on $M \bigcup \{\overline{\ell}\}$ and return its result.

The behavior of the 3-Valued Kripke-Kleene semantics (3KK) is crucial to the efficiency of DP-KK. For mainly this reason, we believe that an investigation into the behavior of 3KK is valuable. In particular we focus on identifying structures in random logic programs which cause 3KK to provide information useful for DP-KK. By identifying regions in the parameter space of our distribution of random logic programs in which the percent of inconsistent models produced by 3KK is large, we are able to characterize properties of the logic programs which cause 3KK to provide information useful for DP-KK. In addition, the behavior of 3KK exhibits enough complexity to be interesting in its own right.

The choice of the distribution of logic programs we use (*the fixed body length model*) is motivated by *the fixed clause-length model* used for random K-SAT instances [9] (we provide only an intuitive definition of the model in this extended abstract and leave a formal one to the full paper). Intuitively, a random logic program is generated by the fixed body length model by repeating the following process m times: uniformly, randomly, and independently choose k letters (with replacement) from the set of n proposition letters and negate each of the last $k-1$ letters independently with probability p. The first letter becomes the head of the rule and the remaining $k-1$ literals become the subgoals. This collection of m rules makes up the random logic program. Similar to the fixed clause length model, we characterize programs generated by the fixed body length model by m/n.

In this extended abstract, unless otherwise stated, we investigate the behavior of 3KK in the fixed body length model by fixing $k=3$, $p=0.5$, and varying n over a range of values investigating, for each, the relationship between m/n and the percentage of inconsistent models produced by 3KK. In all of our experiments, 3KK is computed with respect to a fixed percentage, i_t, of proposition letters (randomly chosen) initially assigned TRUE and a fixed percentage, i_f, (randomly chosen) initially assigned FALSE. For the remainder of this extended abstract, i_t and i_f are fixed at 1 %. In the full paper we also examine the above relationship with i_t, i_f fixed at 5 %.

Throughout our study, we identify when *inference chaining* is required in the making of assignments. Intuitively, a letter ℓ is given an assignment by 3KK requiring inference chaining, if the assignment made to ℓ does not follow entirely from initial assignments or trivial assignments (i.e. assignments made to letters because they do not appear in the heads of any rules) - see section 3 for a formal definition of inference chaining. One of the reasons for classifying assignments as such is that it identifies when 3KK provides information that is not available from immediate inspection of the initial assignments; a second reason is that it reveals the behavior of 3KK more deeply, exposing complexities interesting in their own right.

2 Definitions and the Distribution of Logic Programs

Given a countable set, P, of proposition letters and $a \in P$, a is called an *unnegated* proposition letter and $\overline{a}$ is called a *negated* proposition letter; a *literal* is a negated or an unnegated proposition letter. T denotes the set of all possible literals made up from P. Given $a \in P$, define $letter(a) = letter(\overline{a}) = a$; $letter(\ell)$ is called the proposition letter associated with $\ell \in T$. Given $a \in P$, define $not(a) = \overline{a}$, and $not(\overline{a}) = a$. A propositional logic program, L, is a collection of rules where each rule, R, is a pair $< h, B >$ with $h \in P$ and B a set of literals; h is called the *head* of R, B is called the *body* of R and $b \in B$ is called a *subgoal*. Rules are often written as a string of symbols: $h <- s_1, s_2, \ldots, s_k$ where: $s_1, s_2, \ldots, s_k$ are the subgoals in B and correspond to implication statements; R is read "If s_1, AND s_2, AND ..., AND s_k, Then h". A *partial model*, M, is a subset of T. If $a \in M$ then we say that a is assigned TRUE by M, if $\overline{a} \in M$ then we say that a is assigned FALSE by M, and if $a, \overline{a} \in M$ then we say that a is assigned OVERASSIGNED by M. M is called a *total model* if for every $a \in P$, $a \in M$ or $\overline{a} \in M$. M is called an *inconsistent model* if there exists an $a \in P$ such that $a, \overline{a} \in M$. TRUE and FALSE correspond to a truth value of *true* and *false*, respectively, and OVERASSIGNED to a truth value of both *true* and *false*. A literal, ℓ, is *satisfied* by a partial model, M, if $\ell \in M$ and ℓ is *falsified* by M if $not(\ell) \in M$. The body, B, of a rule, R, is satisfied by a partial model, M, if for all, $b \in B$, b is satisfied by M; B is falsified by M if there exists, $b \in B$, such that b is falsified by M.

We study the behavior of the 3-Valued Kripke-Kleene semantics on logic programs randomly generated according to the *fixed body length model* (see section 1 for a definition). The point of using this model is that the parameters n, m, k, and p tend to characterize the structure of the program generated. For example, programs generated with $m \ll n$ tend to have few rules which "overlap" (i.e. rules whose body contain a subgoal which involves a proposition letter appearing as the head of another rule). Programs generated with $m \gg n$, on the other hand, tend to have many rules which overlap. The ratio m/n affects the amount of overlapping generated programs tend to have. Increasing the value of m/n tends increase the number of rules which overlap in the generated programs, and decreasing m/n tends to decrease the number of rules which overlap in the generated programs.

This model allows us to adjust the structure of the programs generated and study the affects of such adjustments on the behavior of the 3-Valued Kripke-Kleene semantics. For example, by increasing m/n while keeping n fixed we can study how the behavior of 3KK changes as the amount of overlapping amongst the rules increases. The usefulness of this model is not limited to the study of 3KK; due to the interesting results we get in this application it seems that this model might offer a decent testbed for studying the behavior of other semantics (e.g. the well-founded semantics, etc..). Part of our motivation for choosing this model is due to its similarity with the fixed clause length model used for random K-SAT instances [9]. Further, the important role m/n plays in the study of K-SAT motivates our use of m/n as a parameter in the study of 3KK.

3 The 3-Valued Kripke-Kleene Semantics (3KK)

The Kripke-Kleene (KK) semantics was introduced by Fitting [4] as a 3-(or 4-)valued version of Clark's program completion semantics. We modify it — fairly inessentially — in the definition of Φ_L below by adding the final $\cup M$, making Φ_L increasing as well as monotonic.

Let L be a finite propositional logic program over a countable set of proposition letters, P, and T be the set of literals over P. Let $\Phi_L : 2^T \rightarrow 2^T$ defined as follows. For any $M \in 2^T$:

$$\begin{aligned}\Phi_L(M) = \{a \mid a \in P, \exists R =< a, B >\in L \text{ such that: } B \text{ is satisfied by } M\} \bigcup \\ \{\overline{a} \mid a \in P, \forall R =< a, B >\in L \text{ such that: } B \text{ is falsified by } M\} \bigcup M.\end{aligned}$$

Let $M \in 2^T$, M^∞ denotes the least fixed point of Φ_L starting with M. Since Φ_L is monotonic and L finite, then for all $M \in 2^T, M^\infty$ exists (see [4] for a formal exposition). The 3-Valued Kripke-Kleene Semantics (3KK) of L with respect to a partial model, M, is M^∞. Intuitively, 3KK indicates which partial model in the set of all partial models possible for P is chosen to represent the "meaning" of L with respect to M.

An Example

Let $P = \{a, b, c, d, e\}$. Let $M = \{d, \overline{e}\}$. Let L be the logic program represented by:
$e <- \overline{d}, \overline{c}$

$d <\!- \overline{c}, a$
$c <\!- \overline{a}, \overline{b}$

We find the partial model produced by 3KK for L with respect to M. Let $M_1 = \Phi_L(M)$, $M_2 = \Phi_L(M_1), M_3 = \Phi_L(M_2), M_4 = \Phi_L(M_3), \ldots$. Since P in this example is finite, then there exists a least positive integer, α, such that for all positive integers, $z > \alpha, M_\alpha = M_z$. M_α is the least fixed point of Φ_L starting with M so $M^\infty = M_\alpha$. $M_1 = \{\overline{a}, \overline{b}, d, \overline{e}\}$, since a, b do not appear as the head of any rules in L and $d, \overline{e} \in M$. $M_2 = \{\overline{a}, \overline{b}, \overline{c}, \overline{d}, d, \overline{e}\}$, since all of the rules of L with c and d as their heads have bodies falsified by M_1 and $\overline{a}, \overline{b}, d, \overline{e} \in M_1$. $M_3 = \{\overline{a}, \overline{b}, \overline{c}, \overline{d}, d, e, \overline{e}\}$, since e appears in a rule of L whose body is satisfied by M_2 and $\overline{a}, \overline{b}, \overline{c}, \overline{d}, d, \overline{e} \in M_2$. Careful inspection by the reader will show that $M_4 = M_3$ and thus M_3 is the least fixed point of Φ_L starting with M. The 3-Valued Kripke-Kleene Semantics (3KK) of L with respect to M, is M_3 ($M_3 = M^\infty$).

Now we give a definition of inference chaining. Let L be a finite logic program over a countable set, P, of proposition letters and M a partial model; let $M_1 = \Phi_L(M)$. M^∞ is the partial model produced by 3KK on L with respect to M. Let $a \in P$ such that $a \in M^\infty$ or $\overline{a} \in M^\infty$ (i.e. 3KK gives an assignment to a). The assignment given to a by 3KK is said to require *inference chaining* if $M_1 \bigcap \{a, \overline{a}\} \subset M^\infty \bigcap \{a, \overline{a}\}$. In words, the assignment given to a by M^∞ is not the same as the assignment (if any) given to a by M_1.

Intuitively, the assignment made to a does not follow immediately from the initial assignments of M but requires some intermediate assignments to be made to other letters. An assignment is made to a by 3KK for "interesting" reasons if inference chaining is required. The "amount" of inference chaining required of 3KK to produce M^∞ is the number of proposition letters in P which required inference chaining.

The Example Revisited

Now we determine which of the assignments to letters made in M_3 required inference chaining. Since $\overline{b} \in M$, $b \notin M$ and $\overline{b} \in M_3, b \notin M_3$ then the assignment of FALSE given to b by 3KK does not require inference chaining. Since $a, \overline{a} \notin M$ and $\overline{a} \in M_3$ then the assignment of FALSE made to a by 3KK requires inference chaining (notice no assignment is made to a by M but an assignment of FALSE is made to a by M_3). Careful inspection by the reader will reveal that all other assignments made require inference chaining.

4 The Experimental Procedure

We are interested in the behavior of 3KK in terms of the percentage of inconsistent models produced and the amount of inference chaining required to do so. An *experiment* is conducted by fixing a value for m, n, p, k, i_t, and i_f and generating a random logic program, L, according to the fixed body length model and a set of random initial assignments, M_0, according to i_t and i_f. The partial model, M, produced by 3KK on L with respect to M_0 is computed and a variety of information is recorded (consistency, percentage of letters assigned TRUE, etc.). This process is repeated a fixed number of times (number of trials). Throughout all experiments conducted, p, k, i_t, i_f, and number of trials were fixed at 0.5, 3, 1, 1, 100, respectively. We examined the relationship between m/n and the percentage of inconsistent models (with and without inference chaining) and the effect that increasing n had on this relationship.

The results of our study are laid out as follows. Section 5 gives a simple analysis of the limiting behavior in n of 3KK without requiring any inference chaining. Empirical results are presented in this section to describing the behavior of 3KK at "small" values of n. Section 6 presents empirical results describing the behavior of 3KK at "small" values of n.

Briefly summarized, we conclude that as $n \to \infty$ the probability that 3KK produces inconsistent models without requiring inference chaining is 1, but at small n a large region in the parameter space exists where 3KK produces very few inconsistent models. We identify this region and study more closely its causes.

5 Easy Results

This section describes the behavior of 3KK where no inference chaining was required.

5.1 A Simple Analysis

Fix $k = 3$. Let m, n be positive integers, $p \in [0, 1]$, and $i_t, i_f \in [0, 100]$. Let M_0 be a partial model generated according to the procedure of an experiment. Let I_{m,n,p,i_f,i_t} denote the event that, for a logic program generated according to the fixed body length model, the 3-valued Kripke-Kleene semantics of L with respect to M_0 is inconsistent without requiring any inference chaining.

Theorem 1

$\Pr(I_{m,n,p,i_f,i_t}) \geq 1 - (1 - [\frac{i_f}{100} - \frac{1}{n}][\frac{pi_f}{100} + \frac{(1-p)i_t}{100} - \frac{1}{n}]^2)^m$.

Proof: Omitted from this extended abstract.

From Theorem 1 the following Lemma follows directly.

Lemma 1

Given that $i_f, i_t > 0$, and m, n are constrained so that $m/n = c$ for some positive constant c: $lim_{n\to\infty}\Pr(I_{m,n,p,i_f,i_t}) = 1$.

Proof: Follows immediately from Theorem 1.

This result implies that even for very few initial assignments the fixed body length model produces logic programs for which the 3-valued Kripke-Kleene semantics is inconsistent for trivial reasons with high probability as n grows large. In this sense, the fixed body length model is uninteresting in the limiting case for n. However, since logic programs in practice involve a "relatively small" n, we are interested in the behavior of 3KK for small n. Figure 1 gives the percentage of models produced which were inconsistent without requiring any inference chaining. From this figure, we are able to partly fill in the gap between the limiting behavior of the probability that 3KK produces an inconsistent model and the behavior of the

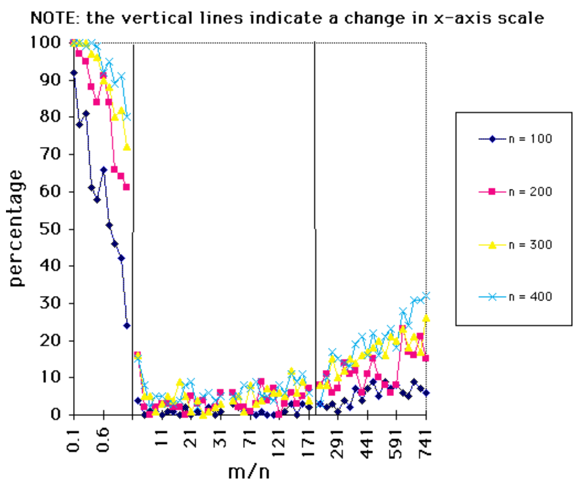

Figure 1: Models inconsistent w/o requiring any inference chaining ($i_t = i_f = 1, p = 0.5$)

probability for small n. We see that there exists a large region in the parameter space where very few inconsistent models are produced without requiring inference chaining. As a result, further study of the behavior of 3KK at small n is justified.

In section 6 we present empirical results describing for small n the percentage of programs for which the 3-valued Kripke Kleene semantics is inconsistent (for non-trivial reasons).

6 Non-Trivial Results

In this section, we continue to fill in the gap between the limiting behavior of the probability that 3KK produces inconsistent models and the probability at small n. We present the results of experiments which show the effect that changing m/n has on the percentage of inconsistent models produced. We observe regions in the parameter space where 3KK produces mostly inconsistent models and a region where 3KK produces almost no inconsistent models. Figure 2 gives the average percentage of models produced by 3KK which were inconsistent (no restriction was placed on the amount of inference chaining 3KK was allowed to make). From Lemma 1, we know that the "dip" will close at large n. From figure 2 we see that the "dip" appears to be closing very slowly. For our purposes, this result is beneficial, since we are interested in the behavior of 3KK at small n. We examine further its behavior at $n = 100$.

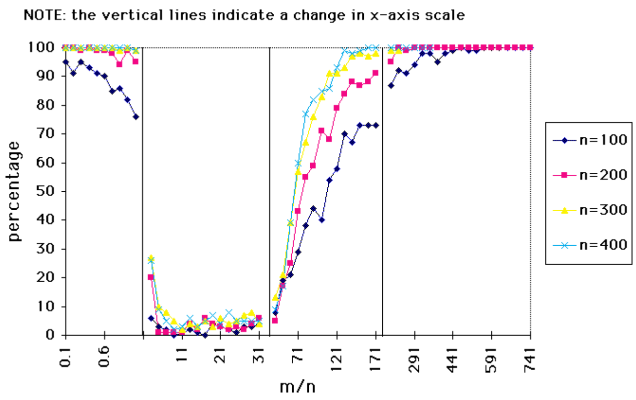

Figure 2: Models inconsistent ($i_t = i_f = 1, p = 0.5$)

Figure 3 gives the average percentage of inconsistent models at $n = 100$ and the percentage without requiring inference chaining. We see that for m/n very small and very large (≤ 1 and ≥ 471) 3KK produces inconsistent models almost always and for intermediate values of m/n (3 - 31), 3KK almost never produces inconsistent models. In addition, a significant amount of inference chaining is required in most cases. This result implies that for $m/n \leq 1$ and $m/n \geq 471$, 3KK is highly effective at pruning branches in the search made by DP-KK and that the branches pruned in most cases were not trivially observed to be prunable (i.e. required a significant amount of inference chaining). Also, the result implies that for $3 \leq m/n \leq 31$, 3KK is ineffective at pruning branches.

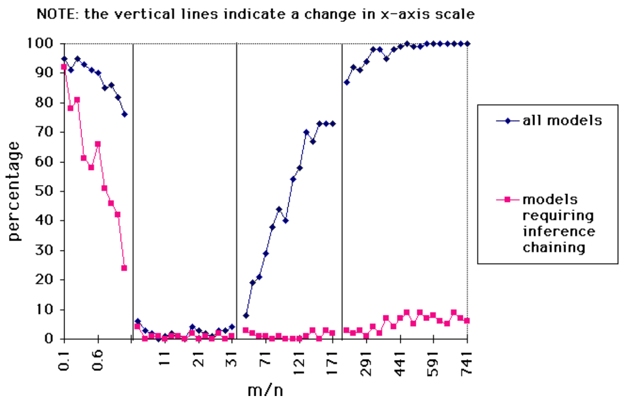

Figure 3: Models inconsistent ($i_t = i_f = 1, p = 0.5, n = 100$)

We examine the behavior of 3KK further in order to understand reasons for its effectiveness and ineffectiveness in DP-KK. Figure 4 gives the average percentage of letters assigned FALSE and TRUE. We discuss the behavior of 3KK at small m/n, large m/n, and intermediate m/n, separately.

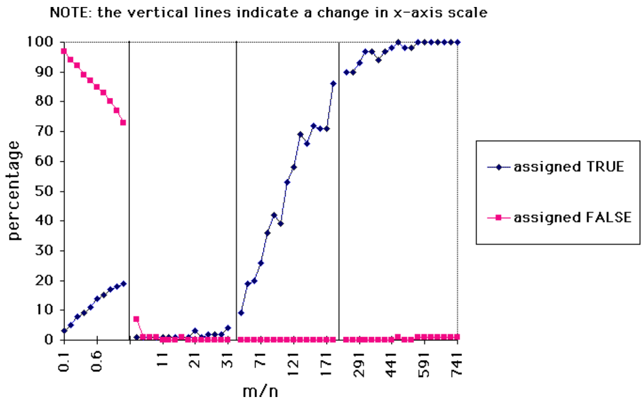

Figure 4: Assignments to letters ($i_t = i_f = 1, p = 0.5, n = 100$)

6.1 Small Values of m/n

Figure 5 gives, for $m/n \leq 3$, the average percentage of inconsistent models at $n = 100$ and the percentage without requiring inference chaining. Figure 6 gives for $m/n \leq 3$ the average percentage of letters assigned FALSE and TRUE. We first offer intuitive explanations for the behavior of the percentage of letters assigned TRUE and FALSE seen in Figure 6 and then using these, provide an intuitive explanation for the behavior of the percentage of inconsistent models seen in figure 5.

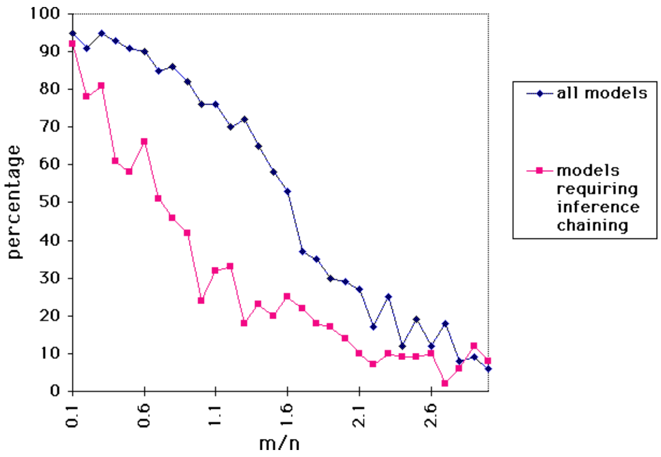

Figure 5: Inconsistent models at small m/n ($i_t = i_f = 1, p = 0.5, n = 100$)

The intuitive explanation for the behavior of the percentage of letters assigned FALSE as seen in Figure 5 is as follows. Since the number of rules in which a letter appears in the head increases with m/n, then the likelihood that a letter will have all of the corresponding bodies falsified by other assignments decreases. Therefore, the percentage of letters assigned FALSE decreases as m/n increases.

We now give an intuitive explanation for the "hump" in the percentage of letters assigned TRUE requiring inference chaining as seen in Figure 5. We conjecture that two opposing effects are causing the hump. The first is the increase in the number of heads in which each letter appears caused by increasing m/n, and the second is the decreasing percentage of letters assigned FALSE as seen in Figure 5.

The first effect causes an increase in the number of letters assigned TRUE by the following mechanism. As the number of heads in which each letter appears increases, the number of letters which appear in the head of a rule whose body is satisfied by the letters assigned FALSE tends to also increase. Thus, the number of letters which are assigned TRUE tends to increase.

The second effect causes a decrease in the number of letters assigned TRUE by the following mechanism. As the percentage of letters assigned FALSE decreases then the number of letters appearing in the head of a rule whose body is satisfied by letters assigned FALSE also decreases. Thus the number of letters assigned TRUE tends to decrease.

We conjecture that for $m/n \leq 1.3$ the first effect dominates causing the percentage of letters assigned TRUE to increase and that for $m/n \geq 1.4$ the second effect dominates causing the percentage TRUE to decrease.

The behavior of the percentage of inconsistent models for small m/n values seen in Figure 6 follows from the behavior of the percentage of letters TRUE and FALSE seen in Figure 5. At very small m/n, the percentage of letters assigned FALSE is very large (> 90), therefore, a small but significant percentage TRUE will cause at least one letter to be assigned OVERASSIGNED (both TRUE and FALSE). As a result, for very small m/n, the percentage of models inconsistent is very large (> 90). However, the percentage of letters assigned FALSE decreases to 7 as m/n increases to 3 and the percentage TRUE remaining small (< 10). As a result the percentage of models inconsistent decreases as m/n increases to 3.

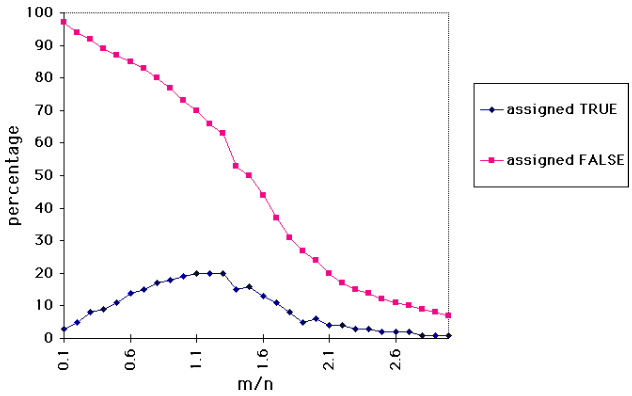

Figure 6: Assignments to letters at small m/n ($i_t = i_f = 1, p = 0.5, n = 100$)

We point out that the "hump" in the percentage of letters assigned TRUE does not seem to affect the percentage of models inconsistent significantly. It seems that the percentage of models inconsistent is relatively insensitive to percentage of letters assigned TRUE.

6.2 Large Values of m/n

From Figure 3 we see that for large values of m/n, the percentage of inconsistent models goes to 100 indicating that in this region 3KK proves very useful for pruning branches in the search for stable models made by DP-KK. From Figure 4 we see the percentage of letters assigned TRUE and FALSE over this region. The percentage of letters assigned FALSE is very small (but not less than 1 since i_f = 1). The reason for this is that at large m/n the number of rules each letter appears in the head of is very large, therefore, the number of letters which appear as the head of only rules whose bodies are falsified is very small.

We also see from Figure 4 that the percentage of letters assigned TRUE increases from 1 to 100 at large m/n. This is due to the fact that since the number of rules in which each letter appear as the head increases with m/n, then the number of letters which appear as the head of at least one rule whose body is satisfied also increases with m/n. The result is that for intermediate sized m/n very few letters appear as the head of at least one rule whose body is satisfied but this number increases with m/n to the point where all letters tend to have at least one corresponding body satisfied.

The percentage of models inconsistent at large m/n seen in Figure 3 seems to follow the percentage of letters assigned TRUE at large m/n as seen in Figure 4. This result is consistent with the percentage of letters which would be assigned OVERASSIGNED if x percent of the letters were randomly and without replacement chosen and assigned FALSE and y percent of the letters were randomly and without replacement chosen and assigned TRUE where x and y are the percentages of letters assigned FALSE and TRUE as seen in Figure 4.

6.3 Intermediate Values of m/n

From Figure 3 we see that for intermediate values of m/n ($5 \leq m/n \leq 41$) a very small percentage of inconsistent models are produced. In this region 3KK is not effective at pruning branches the search for stable models made by DP-KK. From Figure 4 we see that in this region, the percentage of letters assigned TRUE and FALSE is very small. This is due to the following two facts: (1) m/n is large enough to cause letters to appear in the head of enough rules to make the likelihood of one having all its corresponding bodies falsified very small; (2) m/n is not large enough to cause letters to appear in the head of enough rules to prevent the likelihood of one having at least one of its corresponding bodies satisfied from being very small. As a result, in this intermediate range of m/n, very few inconsistent models are produced.

7 Conclusions and Future Work

The main goal of this paper was to point out structural properties that random logic programs generated under the fixed body length distribution have which cause 3KK to produce information useful for guiding the search for stable models made by DP-KK.

7.1 Conclusions

Our conclusions are summarized as follows.

1. For any m/n and $i_f, i_t > 0$, the probability that 3KK produces an inconsistent model without requiring inference chaining tends to 1 as n tends to ∞. However, at $n \leq 400$ there exists large regions in the parameter space where very few inconsistent models are produced without requiring inference chaining.

2. For $i_f = i_t = 1$ and $100 \leq n \leq 400$ (p fixed at 0.5), the models produced at very small values of m/n (≤ 1) are inconsistent with probability at least 0.76 and at very large values of m/n (≥ 471) are inconsistent with probability 1. The models produced at intermediate values of m/n (between 1 and 471) are inconsistent with probability reaching as low as 0.

Conclusion 1 is based on Lemma 1 and Figure 1. Lemma 1 points out that in the limit 3KK always produces inconsistent models for trivial reasons as long as $i_f, i_t > 0$. However, since logic programs in practice may involve only a small number of variables, we are interested in the behavior of 3KK at small n. Figure 1 shows that for small n (≤ 400), a large region exists where very few inferences are made without requiring inference chaining.

Conclusion 2 is based on Figure 2. From this figure we see that for m/n very small (≤ 1) and very large (≥ 471), 3KK produces mostly inconsistent models and, therefore, in these regions, 3KK is very effective at pruning branches in the search for stable models made by DP-KK. We also see from this figure that for intermediate values of m/n (between 1 and 471), 3KK produces a large amount of consistent models indicating that in this region, 3KK is not very effective at pruning branches.

From these conclusions we observe that, for small n, there exist regions in the parameter space of the fixed body length model in which 3KK tends to produce inconsistent models with high probability, therefore, providing information useful to prune branches in the search for stable models made by DP-KK. Also, there exists a region in which, for small n, 3KK tends to produce inconsistent models with a low probability, therefore, providing little information useful for pruning branches in the search made by DP-KK. We conjecture that the "hardest" instances to determine the existence of a stable model occur in this region.

7.2 Future Work

In examining the behavior of 3KK in the fixed body length model, we have six parameters to work with: i_f, i_t, n, m, p, k. In this extended abstract, p and k were fixed at 0.5 and 3, i_t, i_f were fixed at 1 and m, n were constrained to satisfy $m/n = c$ for some constant c. This technique provided one way of examining how structural properties of programs effect the behavior of 3KK, but, there are a variety of other ways in which the parameters could be constrained. Our approach of investigating the effect that increasing n has on the relationship between m/n and the percentage of inconsistent models produced by 3KK provides useful information in the case where n is small, but, for large n, nothing interesting is gained. In some sense this is a weakness of our approach. The limiting behavior of 3KK can provide useful information in certain contexts and therefore a different approach is needed in such contexts.

An example of one such context results from asking the question: does there exists a non-monotonic property which possesses a threshold? Bollobas and Thomason ([1]) prove that any non-trivial monotone increasing property has a threshold in a "weak" sense and experimental evidence exists that the non-trivial monotone increasing property of a random 3-SAT formula having a satisfying truth assignment has a threshold in a "strong" sense ([9] and [8]). Providing experimental evidence that a non-monotonic property has a threshold would be a very interesting result. The property of a logic program having an inconsistent model produced by 3KK with respect to a few initial truth values is non-monotonic in the set of rules. Therefore, providing experimental evidence that this property has a threshold would be a very interesting result. Based on Lemma 1, it seems unlikely that any threshold for this property will be found using the approach we adopted in this paper, but other approaches might yield a threshold.

References

[1] B. Bollobas and A. G. Thomason, Threshold Functions. *Combinatorica 7*, 35-38, 1987

[2] P. M. Dung, 1992 On the relations between stable and well-founded semantics. *Theoretical Computer Science*, 105, 7-25

[3] M. Davis and H. Putnam, A computing procedure for quantification theory, *Journal of the ACM* 7, 1960 pg 201-215

[4] M. Fitting 1985, A Kripke-Kleene Semantics for logic programs. *Journal of Logic Programming*, 2(4), 295-312

[5] M. Fitting 1993, A Family of Stable Models, To Appear in *Journal of Logic Programming*

[6] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming. In *Proceedings of the 5th International Conference Symposium on Logic Programming*, vol. 2, 1070-1080.

[7] W. Marek, A. Nerode, J. Remmel, 1992, How Complicated is the set of stable models of a recursive logic program? *Annals of Pure and Applied Logic*, 56, 119-135

[8] T. Larrabee and Y. Tsuji, Evidence For a Satisfiability Threshold for Random 3CNF Formulas. Technical Report UCSC-CRL-92-42, University of California Santa Cruz, 1992

[9] D. Mitchell, B. Selman, and H. Levesque 1992 Hard and Easy Distributions of SAT Problems *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)* San Jose, CA pg. 459-465

[10] W. Marek, M. Truszczynski, 1991, Autoepistemic Logic *Journal of the ACM*, 38(3), 588-619

[11] A. Van Gelder, K. Ross, J. Schlipf 1991, The Well-Founded Semantics for general logic programs. *Journal of the ACM*, 38(3), pages 620-650