8. Discussion.

This section first considers aspects of the extended-SPASS software that has been developed for this project. Then some features of the axiomatic translation that have not previously been considered are investigated using extended-SPASS. Finally, suggestions for the future development of the project are made.

8.1.1 Is extended-SPASS maintainable?

The code developed for extended-SPASS is designed to be easy to understand, modify, upgrade and debug. It was developed in an evolutionary style. The core features were implemented and tested, and then new features were added and tested, in a series of iterative cycles. It will be easy to add more features to extended-SPASS by following this same pattern of development. Some of the properties of the code that illustrate the good maintainability are discussed below.

The code is fully documented. The description in the theory section (section 2) is tightly coupled to the way in which the software works, and the specification (section 5) has been carefully designed to tie this theory to the output that is produced by extended-SPASS. Using the documentation of the implementation, it is easy to find any particular function(s) in the code responsible for both controlling and performing any particular task. As a result, if new functionality is needed, or a bug is found, then it is easy to locate the code that needs to be changed. During testing, it was rare that the cause of any error was not identified immediately. The code is sufficiently modular and decoupled that modifications or enhancements to any one section do not interfere with other functionality. During debugging, when any error was located, it was invariably confined to a discrete section of code, and when a bug was fixed, no other bugs were caused as a result. The test procedures described in section 7.1 allowed both existing and new features to be tested continually during development.

Software projects frequently have changes in requirements. The ability to accommodate changes in software requirements is a good test of maintainability. Several examples of the way in which modifications have been made are given below.

The schema encodings for a series of new axioms (B^2, B^3, Den, DBBB, G, SR, TR, M, W) were implemented in extended-SPASS after the main code-base had been finished and tested. These were implemented all on the same day, and the mean time for implementation of the schema encoding for each axiom was less than 1 hour. Systematic changes were made to the following functions and data structures: (see the section 6 for further information)

AXIOMTYPE; eml_AllowedSetFlags[]; eml_IsNewAxiom(); eml_AnalyzeSetAxiom(); eml_ProcessLogicX() where X $\in$ { B^2, B^3, Den, DBBB, G, SR, TR, M, W }; eml_ProcessAxiomLogic(); eml_ProcessPreviousComposition(); eml_PrintAxiomComposition(); eml_UpdateCompositionAxiomLogic().

In is apparent that the changes are confined to sections of code that define:

- Processing of the user input from the set_axiom syntax.
- The control structure that defines the mode in which the schema encoding is incorporated into the translation. (These axioms are incorporated in the same way as axiom 5, that is, with compositional subformulae applied to the axiom itself).
- The schema encoding for the new axiom itself, and modifications to the current instantiation set that are related to the new axiom.

Clearly, the code is not as modular as in the object-oriented design for Jasper, but because of the simple and formulaic way in which code needs to be modified, discrete changes can be made rapidly to eml.c to define any new axiom that is required. These changes were made without error, and no disturbance of the pre-existing functions was found during subsequent testing.

In the case of axioms CR, CR2, and CR3, it had not been envisaged at the outset of this project that bi-modal axioms would need to be implemented. The identical modifications just listed were required to support the use of these axioms, together with several other changes. These were modifications to dfgparser.y in order to allow the new syntax for bi-modal axioms to be used in the .dfg file, and a new function eml_SetAxiomFlagValueBi() to support parsing of the new syntax, with minor modifications to the AXIOMFLAG and AXIOM (see section 6.). All the remaining support for bi-modal axioms, for example multiple modality indices in the input formulae was already in place. Again, the changes made were in discrete locations, were formulaic and simple in nature, were implemented without error and did not interfere with pre-existing functionality.

Late in the project the requirements for extended-SPASS were again changed. Schmidt and Hustadt [1] describe the axiomatic translation for local satisfiability calculations, but state that the principles of axiomatic translation can be extended to global satisfiability. Extended-SPASS as used to calculate the results in figures 7.5 to 7.20 (version 1.1.0) implements only local satisfiability. In extended-SPASS version 1.1.2, both local and global satisfiability are available. The changes required to support both global and local satisfiability calculations were made to the following functions and data structures:

- Changes to the `AXIOMHIDDENOPTION` and to the function `eml_AxiomOptionsInit()`, providing two new control flags (`DoNotUseLocalSatisfiability` and `OverrideLocalGlobal`). These flags control the choice of global or local satisfiability translation within the eml-module. The latter is under user control and is interpreted with higher priority.
- Changes to `eml_TranslateToFolMain()` to allow the structure of the `axioms` and `conjectures` lists in the `.dfg` file to influence the choice between local and global satisfiability calculations (by setting `HiddenOpt.DoNotUseLocalSatisfiability`).
- Changes to the function `eml_BuildExistentialTerm()` to implement the choice between local or global satisfiability translation, with the global satisfiability translation defined as $\prod(\varphi) = \forall x\ Q_{\varphi}(x) \wedge \bigwedge\{Def(\psi) \mid \psi \in Sf(\varphi)\}$ (see formula 2.1).

Again, the changes needed to support the new functionality are made in well-defined locations, in a formulaic manner, and are simple in nature. And again, they were implemented without error, tested thoroughly and did not interfere with pre-existing functionality. Again, the maintainability of extended-SPASS has been demonstrated.

8.1.2 Is extended-SPASS flexible and useful?

There are several points to consider when deciding whether extended-SPASS is flexible and useful are discussed briefly below. Some minor problems with the design decisions that were made during this project are also pointed out. These can be rectified in a matter of hours and the reader can expect a modified version of the code incorporating these changes to be available on the project site at the time of reading.

One of the major requirements of this project was to implement the axiomatic translation in SPASS in such a way that the original functioning was not disturbed. The strategy for implementations that was adopted (section 4) appears to have been successful. No examples of disturbance of the functioning to SPASS have come to light. This is an important aspect of the reliability of the software produced. The extensive testing program that was used to check the reliability of the axiomatic translation has been described in section 7.1. It is clear that the different testing mechanism will compensate for shortcomings in the other overlapping test methods, and there is a high degree of confidence that the translations produced for any arbitrary target formula will be correct.

The code produced is portable. In common with the remainder of SPASS, it is written in ANSI C, and has been successfully compiled and tested on five platforms (see section

3). The new web-based interface has been successfully mounted on two platforms and can be accessed by any client with the Firefox 2 browser installed.

Extended-SPASS was developed as a tool for investigating the axiomatic translation. There were no target formulae or axioms, or combinations of axioms that arose in this study, which could not be input and solved with the software. On occasions new aspects of the translation needed to be investigated that were not part of the original requirements (see section 7.1). In each case, changes to the functionality that were made were easily accommodated.

Efficiency is an important consideration. Effort to optimize the translation that is produced by extended-SPASS centered on elimination of unnecessary and duplicated Definition subformulae that would otherwise be submitted to the resolution prover, and duplicate members of the instantiation set. There is some scope still remaining for optimization of the code implementing the actual translation, but since the translation has been shown to be generally very fast, and because most of the improvements that could be made would reduce understandability of the code, they are largely unnecessary.

The web-based interface developed for extended-SPASS was used throughout this project. Of particular use was the ability to debug the `.dfg` script according to the line-number on which problems during execution occurred; to find and copy example scripts to the executable `.dfg` script; and to record the results of execution (together with the script) in the history mechanism.

The command-line user interface has a variety of different input modes (see section 5.1). It remains to be seen which is more popular and useful. The `set_axiom` syntax is the most flexible, with all features of all implemented axioms available. The flag-based syntax is very easy to use in scripted applications (the `set_axiom` feature is more difficult to use because it requires a script to prepare a separate template file for each problem solved).

The project site (see appendix) is able to log access and downloads. To date, the code for extended-SPASS has been downloaded from 8 different countries, and downloads are currently running at more that 15 per week. Although this is a modest number of downloads, it is clear that specialized users are taking an interest.

There are a couple of design decisions made during the development process that make some experiments difficult to run. (These are experiments that were not envisaged at the start of the project). First, axiom 5_o is hard coded so that compositional subformulae update the instantiation set at the same time as the translation incorporates the encodings for axiom 5. In order to perform the experiments described in [1], this formulation is always required to produce a complete translation. However, as described in section 8.2.1

(see figures 8.1.2 and 8.3), it may not always be the case that this formulation of axiom 5 is required. A more flexible way to implement axiom 5, would have been to use the same formulation as the other axioms (like T, D, B, 4, and alt_1; that is, only to update the instantiation set of subsequent axioms during composition). Naturally the default behavior of axiom 5, as already implemented, is correct. In the modified design it is necessary to intervene during the parsing of user input, to convert axiom 5 as requested by the user, by building a compositional sequence of the type 5_o5. This default behavior can then the overridden, to perform the experiments described in section 8.2.1. (Clearly, the same argument can also apply to the new axioms B^k, DBB, DEN, TR and SR, etc). When these were implemented (for section 8.2.1), only small changes were necessary, mainly located in the function `eml_ProcessAxiomLogic()` that controls the processing of the schema encoding, and functions `eml_AnalyzeEmlAxiomTranslationFlags()` and `eml_AxiomOptionsInit()` which pre-process the requested axioms before they are incorporated in the *axiom cache*.

Finally an optimization that has not currently been implemented needs to be considered. Following the axiomatic translation as described in [1] it is only ever necessary to incorporate any particular axiom once into a translation. The experiment described in section 8.2.1 suggests that it may be necessary to instantiate a single axiom several times, for different (overlapping) instantiation sets. As a consequence, duplicate instantiations of a particular schema encoding will occur, leading to duplication of the subformulae submitted to the SPASS resolution prover. Currently, duplicate elimination is implemented only for ‘Definitions’ and members of the instantiation set. If further work confirms that multiple instantiations of a single axiom is important, then this optimization should be implemented, following the same pattern as already established for Definitions.

8.2.1 A candidate for a general complete formulation of the instantiation set for axiom S5 and other combinations of axioms.

In section 7.2, it was observed that a large number of the possible formulations of S5 (and other combinations of axioms) did not yield complete translations for a number of target formulae. These counter-examples are listed in figure 7.10 and 7.11. In section 7.2 is was noted that the instantiation set arising from composition of the component axioms varies according to the formulation, specifically the order of the sequence in which the component axioms are applied. This must be origin of the non-complete formulations.

First, following the pattern suggested in [1], one potential candidate for a generally complete formulation of these composite axioms is simply to define each of the component axioms according to the model set by axiom 5. In this model the compositional subformulae are added to the instantiation set of the axiom itself, and not just to subsequent axioms in the sequence. In extended-SPASS, this formulation can be achieved using the format X_oX in the `set_axiom` syntax, where X is an arbitrary axiom, $X \in \{ T, B, D, 4\}$. (This is the default behavior for axiom 5). The experiment was limited to target formulae that had showed counter-examples in tables 7.10 and 7.11, thus substantially reducing the execution time needed.

Looking at the counter-examples, this hypothesis is doomed to fail. In table 7.10 it was reported that the combination $T5_o$ has counter-examples. There are no compositional subformulae for axiom T, and so the formulation $T5_o$ is identical to T_oT5_o (that is, T_oT5_o5). Nevertheless, as shown in figure 8.1, some interesting results were produced for six of the formulations of other axiom combinations that had previously shown counter-examples. In those cases (5B, T5B, T45, B5D, T45B and D45B), where no counter-examples were seen, it may be that complete translations are produced in for the formulations X_oX. This needs further theoretical investigation and proof. Note that these formulations may represent the minimum number of members of the instantiation set necessary for complete translations. Such formulations are important to maximize the speed at which the SPASS resolution prover is able to solve problems.

However, it is also clear (as expected) that counter-examples exist for some axiom combinations, and so a second, an alternative formulation was examined, in which the instantiation set is manipulated in a different manner. The hypothesis is that if a combination of axioms X is considered where $X \equiv X_1X_2..X_n$ (here X_i is one of B, D, T, 4, 5), then a complete translation is produced by a formulation of the format $X_oX \equiv ((X_1)_o(X_2)_o.. (X_n)_o)X_1X_2..X_n$. In table 8.1.2, the combinations tested are shown. In each case the result produced by the SPASS resolution prover was *compatible with the assertion that this is a general formulation of a complete translation*. Naturally, this remains to be proven. It is worth examining the details of two of these translations more carefully.

This formulation for the combination of axioms 5T is expected to be complete (see table 8.2). Comparison of the translation shows that the same subformulae are present for either 5_oT_o5T or T_o5_oT5, and 5_oT (that is, the translation is the same irrespective of the order of the applied axioms). However, similar data for formulations of 4B is not so promising. The instantiation sets for the formulations 4_oB_o4B or B_o4_oB4 are different, and so a means of defining a translation independent of the order in which the axioms appear has *not* been found (Indeed it may not be possible to generate a formulation in which order is not important in this simple fashion). However, it does appear that these translations may each be complete, and the reason is clear when the data in figure 8.3 is examined in detail. Each translation contains the subformulae `B(□□¬□p) and B(□□p)`, which may therefore be required for completeness (based on their presence in the formulation 4_oB but not in B_o4).

The finding that the second formulation described in figure 8.3 may be complete is interesting; but if this is the case then there are certainly additional subformulae (those which differ between the translations) that can be discarded. A theoretical examination of this formulation needs to be undertaken. The potential importance of this hypothesis is clear. One of the ways in which extended-SPASS will develop in the future is to automatically generate the schema encoding for an arbitrary modal axiom (or at least for an arbitrary member of a class of modal axioms). This will obviate the need to hard-code the encoding of a new modal axioms (although this is not such an onerous task, see section 8.1). It is relatively easy to achieve this automated translation, and a prototype implementation has been made in Jasper (not presented in this dissertation). The use of these encodings in translations that is more challenging (although again, a prototype does exist). In order to make such an extension generally useful, it will be necessary to have a formulation of composite axioms that is *guaranteed to be complete*. To achieve this end, it is necessary to undertake further theoretical work. It is clear that, if a formal proof is required for the completeness of every conceivable combination of a series of arbitrary modal axioms, before the results produced by the tool can be relied upon, then the small amount of time required to implement any new modal axiom manually will pale into insignificance against the time required to generate a formal proof for all the possible ways in which it can be used.

Figure 8.1 <u>Summary of the counter-examples for axiomatic translation of formulae in local satisfiability calculations in various formulations of axiom S5 and other well-know modal axioms</u>.

The test data was calculated for formulae that were identified as counter-examples in figure 7.10 and 7.11. They are:

```
4  4^2  4^3  5  T  aiml02_prop3iii  amai02  amai02b  bbpdddd(qdnp)
bpdddnpg  bpddnpg bqdddddddnq  bqddddddnq  ddddddnqdpbbp  dddddnqbbbq
ddddnqb(qg)  ddddnqbbp  ddddnqbq  ddddnqbq dddnqbq  ddddnqdpbbp  dddnqbbq
dddnqbq  dddnqdpbbp  ddnqbq  demri1  demri2  dnqdbq
```

Figure 8.1.1 <u>Results (for local satisfiability) for axiom formulations of the format X_oX</u>. Formulae that show discrimination between potentially complete and non-complete formulations of the axiom combination are shown (that is, the complete formulation yields the result Proof or Completion, and the incompletion formulation yields the result

Completion or Proof, respectively). Formulations of axiom combinations showing non-complete examples are in brackets. Refer to figure 7.4 for the actual formula.

Axiom Formulation of the format X_oX	**Target Formula for which counter-examples were found**
4_o4B_oB vs $[B_oB4_o4]$	5 ; aiml02_prop3iii ; amai02b ; dnq,dbq
B_oB5_o 5_oB_oB	None
$[TB_oB4_o4$ B_oB4_o4T $B_oBT4_o4]$ vs. $T4_o4B_oB$ 4_o4B_oBT 4_o4TB_oB	5 ; aiml02_prop3iii ; amai02b ; dnq,dbq
$[B_oB4_o4D_oD$ $B_oBD_oD4_o4]$ vs. $D_oDB_oB4_o4$ $D_oD4_o4B_oB$ $4_o4B_oBD_oD$ $4_o4D_oDB_oB$	5 ; aiml02_prop3iii ; amai02b ; dnq,dbq
TB_oB5_o $T5_oB_oB$ 5_oB_oBT 5_oTB_oB B_oBT5_o B_oB5_oT	None
$T4_o45_o$ $T5_o4_o4$ 4_o4T5_o 4_o45_oT 5_oT4_o4 5_o4_o4T	None
$D_oD5_oB_oB$ $D_oDB_oB5_o$ $5_oD_oDB_oB$ $5_oB_oBD_oD$ $B_oB5_oD_oD$ $B_oBD_oD5_o$	None
$T4_o4B_oB5_o$ $T4_o45_oB_oB$ $T5_oB_oB4_o4$ $T5_o4_o4B_oB$ $TB_oB4_o45_o$ $TB_oB5_o4_o4$ $4_o4TB_oB5_o$ $4_o4T5_oB_oB$ $4_o45_oB_oBT$ $4_o45_oTB_oB$ $4_o4B_oBT5_o$ $4_o4B_oB5_oT$ $B_oB4_o4T5_o$ $B_oB4_o45_oT$ $B_oB5_oT4_o4$ $B_oB5_o4_o4T$ $B_oBT4_o45_o$ $B_oBT5_o4_o4$ $5_o4_o4B_oBT$ $5_o4_o4TB_oB$ $5_oTB_oB4_o4$ $5_oT4_o4B_oB$ $5_oB_oB4_o4T$ $5_oB_oBT4_o4$	None
$D_oD4_o4B_oB5_o$ $D_oD4_o45_oB_oB$ $D_oD5_oB_oB4_o4$ $D_oD5_o4_o4B_oB$ $D_oDB_oB4_o45_o$ $D_oDB_oB5_o4_o4$ $4_o4D_oDB_oB5_o$ $4_o4D_oD5_oB_oB$ $4_o45_oB_oBD_oD$ $4_o45_oD_oDB_oB$ $4_o4B_oBD_oD5_o$ $4_o4B_oB5_oD_oD$ $B_oB4_o4D_oD5_o$ $B_oB4_o45_oD_oD$ $B_oB5_oD_oD4_o4$ $B_oB5_o4_o4D_oD$ $B_oBD_oD4_o45_o$ $B_oBD_oD5_o4_o4$ $5_o4_o4B_oBD_oD$ $5_o4_o4D_oDB_oB$ $5_oD_oDB_oB4_o4$ $5_oD_oD4_o4B_oB$ $5_oB_oB4_o4D_oD$ $5_oB_oBD_oD4_o4$	None

Figure 8.1.2 Results (for local satisfiability) for axiom formulations of the format $((X_o)_1(X_o)_2..\ (X_o)_n)X_1X_2..X_n$ For an explanation see 8.1.1. In order to support this experiment, a small modification to extended-SPASS was necessary, as already described in section 8.1.

Axiom Formulation of the format $((X_o)_1(X_o)_2..\ (X_o)_n)X_1X_2..X_n$	**Target Formula for which counter-examples were found**
$T_o4_oB_oT4B$ $T_oB_o4_oTB4$ $B_o4_oT_oB4T$ $B_oT_o4_oBT4$ $4_oT_oB_o4TB$ $4_oB_oT_o4BT$	None
$D_o4_oB_oD4B$ $D_oB_o4_oDB4$ $B_oD_o4_oBD4$ $B_o4_oD_oB4D$ $4_oD_oB_o4DB$ $4_oB_oD_o4BD$	None
$T_oB_o5_oTB5$ $T_o5_oB_oT5B$ $5_oB_oT_o5BT$ $5_oT_oB_o5TB$ $B_oT_o5_oBT5$ $B_o5_oT_oB5T$	None
$T_o4_oB_o5_oT4B5$ $T_o4_o5_oB_oT45B$ $T_o5_oB_o4_oT5B4$ $T_o5_o4_oB_oT54B$ $T_oB_o4_o5_oTB45$ $T_oB_o5_o4_oTB54$ $4_oT_oB_o5_o4TB5$ $4_oT_o5_oB_o4T5B$ $4_o5_oB_oT_o45BT$ $4_o5_oT_oB_o45TB$ $4_oB_oT_o5_o4BT5$ $4_oB_o5_oT_o4B5T$ $B_o4_oT_o5_oB4T5$ $B_o4_o5_oT_oB45T$ $B_o5_oT_o4_oB5T4$ $B_o5_o4_oT_oB54T$ $B_oT_o4_o5_oBT45$ $B_oT_o5_o4_oBT54$ $5_o4_oB_oT_o54BT$ $5_o4_oT_oB_o54TB$ $5_oT_oB_o4_o5TB4$ $5_oT_o4_oB_o5T4B$ $5_oB_o4_oT_o5B4T$ $5_oB_oT_o4_o5BT4$	None
$T_o4_o5_oT45$ $T_o5_o4_oT54$ $4_oT_o5_o4T5$ $4_o5_oT_o45T$ $5_oT_o4_o5T4$ $5_o4_oT_o54T$	None
$D_o4_oB_o5_oD4B5$ $D_o4_o5_oB_oD45B$ $D_o5_oB_o4_oD5B4$ $D_o5_o4_oB_oD54B$ $D_oB_o4_o5_oDB45$ $D_oB_o5_o4_oDB54$ $4_oD_oB_o5_o4DB5$ $4_oD_o5_oB_o4D5B$ $4_o5_oB_oD_o45BD$ $4_o5_oD_oB_o45DB$	None

$4_oB_oD_o5_o4BD5$ $4_oB_o5_oD_o4B5D$ $B_o4_oD_o5_oB4D5$ $B_o4_o5_oD_oB45D$ $B_o5_oD_o4_oB5D4$ $B_o5_o4_oD_oB54D$ $B_oD_o4_o5_oBD45$ $B_oD_o5_o4_oBD54$ $5_o4_oB_oD_o54BD$ $5_o4_oD_oB_o54DB$ $5_oD_oB_o4_o5DB4$ $5_oD_o4_oB_o5D4B$ $5_oB_o4_oD_o5B4D$ $5_oB_oD_o4_o5BD4$	
$D_o5_oB_oD5B$ $D_oB_o5_oDB5$ $5_oD_oB_o5DB$ $5_oB_oD_o5BD$ $B_o5_oD_oB5D$ $B_oD_o5_oBD5$	None
4_oB_o4B B_o4_oB4	None
B_o5_oB5 5_oB_o5B	None
5_oT_o5T T_o5_oT5	None

Table 8.2 <u>An example of the instantiation set for various formulations of the axiom combination 5T</u>. The translation of the target formula $\Box p\wedge\neg\Box\Box p$ ('4' in figure 7.4), with base instantiation set { $\Box p\wedge\neg\Box\Box p$, $\Box p$, $\Box\Box p$, p }, is shown. (For an explanation of the notation used, see section 2). Formulation $T5_o$ is not complete, but the other two formulations are candidates for a complete translation (in local satisfiability). Note that a simple shell script is able to extract the contents of this table from the information printed by extended-SPASS. (In the formulations 5_oT_o5T, T_o5_oT5 and T_o5_oT, 5_o is not shorthand for 5o5, as it is elsewhere).

5_oT_o5T	T_o5_oT5 or T_o5_oT	$T5_o$	5_oT
Def($\Box\Box p$)	Def($\Box\Box p$)	Def($\Box\Box p$)	Def($\Box\Box p$)
Def($\Box p$)	Def($\Box p$)	Def($\Box p$)	Def($\Box p$)
Def(p)	Def(p)	Def(p)	Def(p)
Def($\Box p\wedge\neg\Box\Box p$)	Def($\Box p\wedge\neg\Box\Box p$)	Def($\Box p\wedge\neg\Box\Box p$)	Def($\Box p\wedge\neg\Box\Box p$)
5($\Box\Box p$)	T($\Box\Box p$)	T($\Box\Box p$)	5($\Box\Box p$)
5($\Box p$)	T($\Box p$)	T($\Box p$)	5($\Box p$)
5($\Box\neg\Box p$)	5($\Box\Box p$)	5($\Box\Box p$)	5($\Box\neg\Box p$)
5($\Box\neg\Box\Box p$)	5($\Box p$)	5($\Box p$)	5($\Box\neg\Box\Box p$)
Def($\Box\neg\Box p$)	5($\Box\neg\Box p$)	5($\Box\neg\Box p$)	Def($\Box\neg\Box p$)
Def($\Box\neg\Box\Box p$)	5($\Box\neg\Box\Box p$)	5($\Box\neg\Box\Box p$)	Def($\Box\neg\Box\Box p$)
T($\Box\Box p$)	Def($\Box\neg\Box p$)	Def($\Box\neg\Box p$)	T($\Box\Box p$)
T($\Box p$)	Def($\Box\neg\Box\Box p$)	Def($\Box\neg\Box\Box p$	T($\Box p$)
T($\Box\neg\Box p$)	T($\Box\neg\Box p$)		T($\Box\neg\Box p$)
T($\Box\neg\Box\Box p$)	T($\Box\neg\Box\Box p$)		T($\Box\neg\Box\Box p$)

Table 8.3 <u>An example of the instantiation set for various formulations of the axiom combination 4B</u>. Translations of the target formula $\neg\Box\neg\Box p\wedge\neg\Box p$ ('5' in figure 7.4), with base instantiation set { p, $\Box p$, $\Box\neg\Box p$, $\neg\Box\neg\Box p\wedge\neg\Box p$ } in a variety of formulations of 4B is shown, together with the results from the resolution prover for a calculation of local satisfiability. (For an explanation of the notation used, see section 2). In section 7.1 it has already been seen that the formulation 4_oB may be complete.

4B	4_oB	B_o4	4_oB_o4	4_oB_o4B	B_o4_oB	B_o4_oB4
Completion	Proof	Completion	Proof	Proof	Proof	Proof
Def($\Box p$)	Def($\Box p$)	Def($\Box p$)	Def($\Box p$)	Def($\Box p$)	Def($\Box p$)	Def($\Box p$)
Def($\Box\neg\Box p$)	Def($\Box\neg\Box p$)	Def($\Box\neg\Box p$)	Def($\Box\neg\Box p$)	Def($\Box\neg\Box p$)	Def($\Box\neg\Box p$)	Def($\Box\neg\Box p$)
Def($\neg\Box\neg\Box p\wedge\neg\Box p$)	Def($\neg\Box\neg\Box p\wedge\neg\Box p$)	Def($\neg\Box\neg\Box p\wedge\neg\Box p$)	Def($\neg\Box\neg\Box p\wedge\neg\Box p$)	Def($\neg\Box\neg\Box p\wedge\neg\Box p$)	Def($\neg\Box\neg\Box p\wedge\neg\Box p$)	Def($\neg\Box\neg\Box p\wedge\neg\Box p$)
Def(p)	Def(p)	Def(p)	Def(p)	Def(p)	Def(p)	Def(p)

4(□p)
4(□¬□p)
B(□p)
B(□¬□p)
	4(□p)
	4(□¬□p)
	Def(□□¬□p)
	Def(□□p)
	B(□p)
	B(□¬□p)
	B(□□¬□p)
	B(□□p)
		B(□p)
		B(□¬□p)
		Def(□¬□¬□p)
		4(□p)
		4(□¬□p)
		4(□¬□¬□p)
			4(□p)
			4(□¬□p)
			Def(□□¬□p)
			Def(□□p)
			B(□p)
			B(□¬□p)
			B(□□¬□p)
			B(□□p)
			Def(□¬□□p)
			Def(□¬□□¬□p)
			Def(□¬□¬□p)
			4(□□¬□p)
			4(□□p)
			4(□¬□□p)
			4(□¬□□¬□p)
			4(□¬□¬□p)
				4(□p)
				4(□¬□p)
				Def(□□¬□p)
				Def(□□p)
				B(□p)
				B(□¬□p)
				B(□□¬□p)
				B(□□p)
				Def(□¬□□p)
				Def(□¬□□¬□p)
				Def(□¬□¬□p)
				4(□□¬□p)
				4(□□p)
				4(□¬□□p)
				4(□¬□□¬□p)
				4(□¬□¬□p)
				B(□¬□□¬□p)
				B(□¬□¬□p)
				B(□¬□□p)
					B(□p)
					B(□¬□p)
					Def(□¬□¬□p)
					4(□p)
					4(□¬□p)
					4(□¬□¬□p)
					Def(□□¬□¬□p)
					Def(□□¬□p)
					Def(□□p)
					B(□¬□¬□p)
					B(□□¬□¬□p)
					B(□□¬□p)
					B(□□p)
						B(□p)
						B(□¬□p)
						Def(□¬□¬□p)
						4(□p)
						4(□¬□p)
						4(□¬□¬□p)
						Def(□□¬□¬□p)
						Def(□□¬□p)
						Def(□□p)
						B(□¬□¬□p)
						B(□□¬□¬□p)
						B(□□¬□p)
						B(□□p)
						4(□□¬□¬□p)
						4(□□¬□p)
						4(□□p)

8.2.2 <u>A brief examination of the complexity of the axiomatic translation.</u>

The axiomatic translation is very prolific in introducing new subformulae into the final formula submitted to the SPASS resolution prover. The number of subformulae submitted to the resolution prover for any particular translation is not easy to calculate. It depends upon the axiom combination, whether compositional subformulae are included or not, whether shortcuts are included, and the instantiation set arising from the target formula. The information described below and seen in table 8.4 can be used to determine this quantity. In this table, no optimizations, like elimination of duplicate Definitions, are considered and so the quantity calculated is for the ‘worst case’.

Take for example the translations illustrated in figure 8.3. First, it is necessary to define some notation. $|\psi|$ is the number of members of the base instantiation set; $|\Box\psi|$ is the number of members of the current instantiation set with a leading box symbol. The number *elements* in the translation is defined in terms of the newly defined predicates that are introduced into the calculation. An *element* is an instance (or occurrence) of a predicate symbol (Q or R), which is submitted in the `FINAL TERM` to the SPASS resolution prover. All instances of these predicates, contributed by either Definition of the new predicates or by encoding of the axiom schema, are counted by a function μ that takes as arguments the target formula φ and the sequence of axioms (*A* in the sequence Δ)

Number of instances of predicates submitted to resolution, that is,

number of *elements* $=$ $\mu(\phi, \{A \mid A \in \Delta\})$.

The three components contributing to this count are listed below,

$$\mu(\phi, \{A \mid A \in \Delta\}) = \mu_1 + \bigwedge_{\{A \mid A \in \Delta\}}(\mu_2 + \mu_3)$$

The last two of these components are defined in terms of the values Quantity 1 to Quantity 4. Broadly μ_2 refers to the number of elements introduced directly by the schema encoding itself, and μ_3 refers to new Definitions arising as a result of addition of new members to the current instantiation set as a result of composition of axioms. For details, see below. For an explanation of the values Quantity 1 to Quantity 4, see table 8.4.

The complexity of Definition (formulae 2.1 to 2.4) is not simple to calculate. First is depends on the number of members in the base instantiation set. For each of these, it then depends on the structure of the instantiation set member being encoded, for example, because the number of elements arising in the π transformation varies (formulae 2.3). If shortcuts are included, then at least 4 elements arise (the special case for a predicate in formula 2.4), and for the common formulations of instantiation set members (in formulae 2.3) there will be 8-10 elements introduced into the calculation. Instantiation set members of the type seen in formula 2.3.1 will give rise to even more elements. (In each of these cases there is an extra element introduced, comprising the leading subformula in formula 2.1, but since this is common to all translations it can be ignored). It is worth noting that when an instantiation set member with a leading box symbol is Defined, then 10 elements are always introduced into the calculation. (Taking the simplest example, the Definition of $\Box(r,p)$, in formulae 2.2 (see also formulae 2.3), gives a total of 10 elements as follows:

the second line contributes 2 elements: $\forall x(Q_{\Box r,p}(x)\rightarrow\neg Q_{\neg\Box r,p}(x))$;

and if shortcuts are included, another 2 elements: $\forall x(Q_{\Box r,p}(x)\leftarrow\neg Q_{\neg\Box r,p}(x))$;

the first line contributes 3 elements: $\forall x(Q_{\Box r,p}(x)\rightarrow\forall y(R_r(x,y)\rightarrow Q_p(y)))$;

and the third line contributes 3 elements: $\forall x(Q_{\neg\Box r,p}(x)\rightarrow\exists y(R_r(x,y)\wedge Q_{\neg p}(y)))$).

Explanation of μ_1. The number of elements formed by translation of the target formula needs to be considered. This value is built up by considering the number of elements introduced by Definition of each of the members of the base instantiation set. For all the translations in figure 8.3, the translation (Definition) of the target formula gives rise to 34 elements. (These are the elements in the fragments `Def(□p)`, `Def(□¬□p)`, `Def(¬□¬□p∧¬□p)` and `Def(p)`).

When the incorporation of an axiom is considered, the number of elements introduced can be calculated by considering the values given in table 8.4. In each of the following cases, each axiom needs to be considered in turn, and the result is summed over all axioms in the sequence (giving rise to the value $\bigwedge_{\{A\mid A\in\Delta\}}(\mu_2+\mu_3)$).

Explanation of μ_2. First the number of elements introduced directly by the schema encoding itself, and by the new predicate symbols that need to be Defined to support the schema encoding is given by:

$$(|\Box\psi|)*\text{Quantity1} + (|\Box\psi|)*\text{Quantity2} \qquad \text{[formula 8.1]}$$

Briefly, Quantity1 deals with the number of elements arising in the schema encoding and Quantity2 deals with the new predicate symbols that need to be Defined (where the number of elements per Definition is 10). In table 8.3 for the formulation 4_oB, this equation refers to the elements in the following fragments: for axiom 4 fragments `4(□p)` and `4(□¬□p)` giving rise to a total of 6 elements, and for axiom B fragments `B(□p)`, `B(□¬□p)`, `B(□□p)`, `B(□□¬□p)` giving rise to a total of 12 elements. Of course, there are no new predicate symbols that need to be Defined for axioms 4 and B.

At the next axiom in the sequence of axioms, and only if composition is being considered, the current instantiation set is updated with the following number of new members:

$$|\Box\psi|_{new}=(|\Box\psi|)*\text{Quantity3} + (|\Box\psi|) \qquad \text{[formula 8.2]}$$

In Table 8.3 for the formulation 4_oB, this equation refers to the two new members of the instantiation set, `□□p` and `□□¬□p`, arising from composition of axiom 4_o, giving a total of 4 members of $\Box\psi$.

Explanation of μ_3. Since these new members of the instantiation set are later incorporated into an axiom schema, they each need to be Defined, giving rise to the following number of elements:

$$(|\Box\psi_{previous}|)*\text{Quantity4} \qquad \text{[formula 8.3]}$$

In Table 8.3 for the formulation 4_oB, this equation refers to the fragments `Def(□□p)` and `Def(□□¬□p)`, and $|\Box\psi_{previous}|$ = 4-2 = 2, giving rise to a total of 20 elements.

Two examples are now given to illustrate the use of the data in table 8.4.

First, for translation of the target formula in axioms 4B (without composition), the number of elements is given by:

- For the translation of the target formula (μ_1): 34 elements

  (see above; `Def(□p)`, `Def(□¬□p)`, `Def(¬□¬□p∧¬□p)` and `Def(p)`).
- For axiom 4 with $\Box\psi$={`□p`, `□¬□p`} and $|\Box\psi|$=2 (μ_2):

  (2*3) + (2*0) = 6 elements

  (see formula 8.1; referring to fragments `4(□p)` and `4(□¬□p)`, and using Quantities 1 and 2 for axiom 4).
- For axiom B with $\Box\psi$={`□p`, `□¬□p`} and $|\Box\psi|$=2 (μ_2):

  (2*3) + (2*0) = 6 elements

  (see formula 8.1; referring to fragments `B(□p)`, `B(□¬□p)`, and using Quantities 1 and 2 for axiom B).
- Total: = 46 elements.

Likewise in table 8.3 for the translation of the target formula in $4_o B_o 4B$, the number of elements is given by:

- For the translation of the target formula (μ_1): 34 elements

  (see above; referring to fragments `Def(□p)`, `Def(□¬□p)`, `Def(¬□¬□p∧¬□p)` and `Def(p)`).

- For axiom 4_o with □ψ={`□p`, `□¬□p`} and |□ψ|=2 (μ_2):

  (2*3) + (2*0) = 6 elements

  (see formula 8.1; referring to fragments `4(□p)` and `4(□¬□p)`, and using Quantities 1 and 2 for axiom 4).

- Update of the instantiation set with □ψ={`□p`, `□¬□p`} and |□ψ|=2 for composition of axiom 4_o gives the new total size of the instantiation set as:

  $|\Box\psi|_{new}$= 2 + (2*1) = 4 members

  (see formula 8.2; referring to new members `□□p` and `□□¬□p`, and using Quantity 3 for axiom 4)).

- For axiom B_o with □ψ={`□p`, `□¬□p`, `□□p`, `□□¬□p` } and |□ψ|=4 (μ_2):

  (4*3) + (4*0) = 12 elements

  (see formula 8.1; `B(□p)`, `B(□¬□p)`, `B(□□p)` and `B(□□¬□p)`, and using Quantities 1 and 2 for axiom B).

- The new predicate symbols introduced by composition of axiom 4 need to be Defined (using the difference between the old and new values for |□ψ|= 4-2 = 2) (μ_3):

  2*10 = 20 elements

  (see formula 8.3; referring to fragments `Def(□□p)` and `Def(□□¬□p)`, and using Quantity 4 for axiom 4).

- Update of the instantiation set with |□ψ|= {`□ p`, `□¬□p`, `□□p`, `□ □¬□p`} and |□ψ|=4 for composition of axiom B_o gives the new total size of the instantiation set as:

  $|\Box\psi|_{new}$= 4 + (4*1) = 8 members

  (see formula 8.2; referring to new members `□¬□p`, `□¬□¬□p`, `□¬□□p`, `□¬□□¬□p`, and using Quantity 3 for axiom B).

- For axiom 4 with □ψ={ `□p`, `□¬□p`, `□□p`, `□□¬□p`, `□¬□p`, `□¬□¬□p`, `□¬□□p`, `□¬□□¬□p` } and |□ψ|=8 (μ_2):

  (8*3) + (8*0) = 24 elements

  (see formula 8.1; referring to fragments `4(□p)`, `4(□¬□p)`, `4(□□p)`, `4(□□¬□p)`, `4(□¬□p)`, `4(□¬□¬□p)`, `4(□¬□□p)`, `4(□¬□□¬□p)`, and using Quantities 1 and 2 for axiom 4).

- The new predicate symbols introduced by composition of axiom B need to be Defined (using the difference between the old and new values for $|\Box\psi| = 8-4 = 4$) (μ_3):

  $4*10 = 40$ elements

  (see formula 8.3; referring to fragments `Def(□¬□p)`, `Def(□¬□¬□p)`, `Def(□¬□□p)`, `Def(□¬□□¬□p)`, and Quantity 4 for axiom B).
- For axiom B with `□ψ={ □p, □¬□p, □□p, □□¬□p, □¬□p, □¬□¬□p, □¬□□p, □¬□□¬□p }` and $|\Box\psi|=8$ (μ_2):

  $(8*3) + (8*0) = 24$ elements

  (see formula 8.1; referring to fragments `B(□p)`, `B(□¬□p)`, `B(□□p)`, `B(□□¬□p)`, `B(□¬□p)`, `B(□¬□¬□p)`, `B(□¬□□p)`, `B(□¬□□¬□p)`, and using Quantities 1 and 2 for axiom B).
- Total: = 160 elements.

These examples illustrate that the expansion of the current instantiation set, as more axiom schema are instantiated in compositional mode, is the source of an explosion in the number of elements introduced into the resolution calculation. Clearly, optimization by elimination of duplicate elements would significantly reduce the quantity of elements introduced (the translations seen in figure 8.3 are optimized, and the translation in 4_oB_o4B produces only 126 elements). Even without composition, a large number of elements are still introduced by the instantiation axiom schemas, so it is easy to see why correspondence properties might be lead to a simpler calculation when introduced into the formula submitted to resolution. This examination of complexity is quite simple. Clearly these calculations could be extended to take account of the different predicate types (accessibility relation R, and other predicates Q), which are treated in a very different way by the resolution prover.

Figure 8.4 **Information required to estimate the number of subformulae introduced into any translation by incorporation of an axiom**.

The table shows a digest of tables 2.8 and 2.18, together with 4 quantities required to estimate the number of elements introduced into the translation by each instantiation of a schema encoding for an axiom. □ψ is the current instantiation set edited for members with a leading □ symbol (like □p). All variables in the table are universally quantified, unless otherwise stated. The quantities quoted are given in *elements*, as explained in section 8.2.2. The meaning of these 4 quantities are described below:

Quantity 1: The number of *elements* introduced by instantiation of the axiom schema (per member of □ψ).

Quantity 2: The number of *elements* introduced by Defintion of new predicate symbols introduced during the schema encoding considered in Quantity 1 (per member of $\Box\psi$).

Quantity 3: The number of *members* added to the instantiation set as a result of composition by the schema encoding considered in Quantity 1 (per member of $\Box\psi$).

Quantity 4: The number of elements introduced by Definition of new Predicate symbols introduced in Quantity 3 (when they are encoded in subsequent axioms; again per member of the instantiation set $\Box\psi$).

Note, no optimization by elimination of duplication is considered, and so for example axiom 5_o is considered as 5_o5.

Axiom	**T**	**B**	**D**	**4**	**5**	**alt$_1$**
Modal axiom	$\neg\Box p\vee$ p	$\Box\neg\Box p\vee$ p	$\neg\Box p\vee$ $\neg\Box\neg p$	$\neg\Box p\vee$ $\Box\Box p$	$\Box\neg\Box p\vee$ $\Box p$	$\Box p\vee$ $\Box\neg p$
Schema clauses	$\neg Q_{\Box p}(x)\vee$ $Q_p(x)$	$\neg R(x,y)\vee$ $\neg Q_{\Box p}(y)\vee$ $Q_p(x)$	$\neg Q_{\Box p}(x)\vee$ $Q_{\neg\Box\neg p}(x)$	$\neg Q_{\Box p}(x)\vee$ $\neg R(x,y)\vee$ $Q_{\Box p}(y)$	$\neg R(x,y)\vee$ $\neg Q_{\Box p}(y)\vee$ $Q_{\Box p}(x)$	$\neg Q_{\neg\Box p}(x)\vee$ $Q_{\Box\neg p}(x)$
Quantity 1…	2	3	2	3	3	2
Schema symbols Defined	–	–	$\Box\neg p$	–	–	$\Box\neg p$
Quantity 2…	–	–	10*1	–	–	10*1
Composition instantiation set	–	$\Box\neg\Box p$	$\Box\neg p$	$\Box\Box p$	$\Box\neg\Box p$	$\Box\neg p$
Quantity 3…	–	1	1	1	1	1
Compositional symbols Defined	–	$\Box\neg\Box p$	–	$\Box\Box p$	$\Box\neg\Box p$	–
Quantity 4…	–	10*1	–	10*1	10*1	–

Axiom	**4^2**	**4^3**	**5^2**	**5^3**	**alt$_1^{1,1}$**	**alt$_1^{1,2}$**	**alt$_1^{2,1}$**	**alt$_1^{2,2}$**
Modal axiom	$\neg\Box p\vee$ $\Box\Box\Box p$	$\neg\Box p\vee$ $\Box\Box\Box\Box p$	$\Box\Box\neg\Box p\vee$ $\Box p$	$\Box\Box\Box\neg\Box p\vee$ $\Box p$	$\Box\Box p\vee$ $\Box\Box\neg p$	$\Box\Box p\vee$ $\Box\Box\Box\neg p$	$\Box\Box\Box p\vee$ $\Box\Box\neg p$	$\Box\Box\Box p\vee$ $\Box\Box\Box\neg p$
Schema clauses	$\neg Q_{\Box p}(x)\vee$ $\neg R(x,y)\vee$ $\neg R(y,z)\vee$ $Q_{\Box p}(z)$	$\neg Q_{\Box p}(x)\vee$ $\neg R(x,y)\vee$ $\neg R(y,z)\vee$ $\neg R(z,u)\vee$ $Q_{\Box p}(u)$	$Q_{\Box p}(x)\vee$ $\neg R(x,z)\vee$ $\neg R(z,y)\vee$ $\neg Q_{\Box p}(y)$	$Q_{\Box p}(x)\vee$ $\neg R(x,z)\vee$ $\neg R(z,u)\vee$ $\neg R(u,y)\vee$ $\neg Q_{\Box p}(y)$	$\neg R(x,y)\vee$ $\neg Q_{\neg\Box p}(y)\vee$ $\neg R(x,z)\vee$ $Q_{\Box\neg p}(z)$	$\neg R(x,y)\vee$ $\neg Q_{\neg\Box p}(y)\vee$ $\neg R(x,u)\vee$ $\neg R(u,z)\vee$ $Q_{\Box\neg p}(z)$	$\neg R(x,u)\vee$ $\neg R(u,y)\vee$ $Q_{\neg\Box p}(y)\vee$ $\neg R(x,z)\vee$ $Q_{\Box\neg p}(z)$	$\neg R(x,u)\vee$ $\neg R(u,y)\vee$ $\neg Q_{\neg\Box p}(y)\vee$ $\neg R(x,v)\vee$ $\neg R(v,z)\vee$ $Q_{\Box\neg p}(z)$
Quantity 1…	4	5	4	5	4	5	5	6
Schema symbols Defined	–	–	–	–	$\Box\neg p$	$\Box\neg p$	$\Box\neg p$	$\Box\neg p$
Quantity 2…	–	–	–	–	10*1	10*1	10*1	10*1
Composition Instantiation Set	$\Box\Box\Box p$ $\Box\Box p$	$\Box\Box\Box\Box p$ $\Box\Box\Box p$ $\Box\Box p$	$\Box\Box\neg\Box p$ $\Box\neg\Box p$	$\Box\Box\Box\neg\Box p$ $\Box\Box\neg\Box p$ $\Box\neg\Box p$	$\Box\Box p$ $\Box\Box\neg p$	$\Box\Box p$ $\Box\Box\Box\neg p$ $\Box\Box\neg p$	$\Box\Box\Box p$ $\Box\Box\neg p$ $\Box\Box p$	$\Box\Box\Box p$ $\Box\Box\Box\neg p$ $\Box\Box\neg p$ $\Box\Box p$
Quantity 3…	2	3	2	3	2	3	3	4
Compositional symbols Defined	$\Box\Box\Box p$ $\Box\Box p$	$\Box\Box\Box\Box p$ $\Box\Box\Box p$ $\Box\Box p$	$\Box\Box\neg\Box p$ $\Box\neg\Box p$	$\Box\Box\Box\neg\Box p$ $\Box\Box\neg\Box p$ $\Box\neg\Box p$	$\Box\Box p$ $\Box\Box\neg p$	$\Box\Box p$ $\Box\Box\Box\neg p$ $\Box\Box\neg p$	$\Box\Box\Box p$ $\Box\Box\neg p$ $\Box\Box p$	$\Box\Box\Box p$ $\Box\Box\Box\neg p$ $\Box\Box\neg p$ $\Box\Box p$
Quantity 4…	10*2	10*3	10*2	10*3	10*2	10*3	10*3	10*4

8.3 Further work.

This study has produced a tool for investigating the axiomatic translation of modal logic into first-order logic. The use of the tool, as described in this dissertation, has suggested a good deal of further work, which it has not been possible to complete in the limited time available (and was not part of the project specification). Throughout the last two chapters suggestions have been made of further work. Additional points are raised below.

In this study many formulations of S5 have been screened and many have been found likely to be complete. However no proofs were offered. This theoretical work needs to be done. In a similar way, axiomatic translations for axioms not covered in [1] have been suggested (axioms B^2, B^3, Den, DBBB, G, S TR, CR, CR2, CR3), but not proven.

The object-oriented implementation of the axiomatic translation in Java (Jasper) is interesting. For the purposes of this study Jasper was formulated as a test system. However it is worth considering a full implementation of a theorem prover (not just the axiomatic translation) in Java, and it is certainly worth fully implementing the current axiomatic translation (and any further developments) in Jasper. In the later case, it would be possible to provide an interface to the existing SPASS theorem prover. (Interfaces like JNI are available in Java, and are commonly used in applications like database access). One of the reasons that Java is often dismissed as a language for implementation of compute-intensive applications is the perception that the speed of execution will suffer greatly. Since 1996, it has been recognized that the implementation of compute-intensive code in Java (versus C) seldom results in an overall slowdown of the application [33], and the situation today is better, with better compilers, etc. It is worth noting that in tests, the equivalent axiomatic translations in Jasper were faster than in extended-SPASS. The design, documentation, implementation, maintainability and testing advantages of an object-oriented language over C are very well known and need not be repeated here.

One type of axioms was not considered in this study; n-ary axioms, with more than one free propositional variable. An example would be Hintikka: $\Box(\Box p \rightarrow q) \vee \Box(\Box q \rightarrow p)$.

This has a proposed axiomatic schema encoding of

$$\forall x(\forall y(\neg R(x,y) \vee \neg Q_{\Box p}(y) \vee Q_q(y)) \vee \forall z(\neg R(x,z) \vee \neg Q_{\Box q}(z) \vee Q_p(z))),$$

and is testable using the correspondence property [26]

$$\forall xyz((R(x,y) \wedge R(x,z)) \rightarrow (R(y,z) \vee R(z,y)))$$

It may be a good idea to modify extended-SPASS to enable it to perform a translation in this (and related) axioms.

Finally, some effort was made in this study to investigate the influence of *precedence* on the efficiency of the axiomatic translation. The precedence adopted in the translation is described in section 5.7. This follows other work to some extent [34]. The highest priority relationship in the precedence scheme used makes accessibility relation symbols (R) strictly greater than other predicate symbols (Q). This is required to produce a decidable translation in ordered resolution. The other features of the precedence (described in section 5.7) are designed to increase the speed in which a solution is found during resolution (reduce the search space), but also to provide a total ordering of the newly introduced symbols. When the precedence is correctly set, a solution will be found earlier, since calculations that are likely to yield a result are performed first. A brief search of different precedence schemes (these experiments are lengthy) showed that the scheme adopted in section 5.7 was the best. Other schemes that were tested implemented only a subset of the elements used in the final scheme. The major improvements were seen in cases where the outcome of resolution was Proof Found (unsatisfiable), which tend to be solved more quickly (see section 7.2). Briefly, it was reasoned that the less commonly occurring symbols are more likely to yield a result in resolution, and hence they were given higher precedence. After that, the precedence is arranged so that sub-formulae (of various types) are given a lower precedence than super-formulae. After that, the remaining ordering is arbitrary, and serves largely to ensure that a total ordering is achieved. It seems clear that precedence is a likely to be a very fruitful means of increasing the efficiency of resolution. A large amount of recent literature exists in which different schemes have been developed, and implemented [35].