

7. Testing and Results.

7.1 Software Testing.

Software testing was conducted in five phases:

- Unit testing.
- Manual inspection of output of extended-SPASS.
- Comparison of the output TERMS from the axiomatic translation of a set of target formulae in extended-SPASS, to the same results generated by an independent implementation of the axiomatic translation in Java (Jasper).
- Comparison of the final results of resolution (Proof/Completion found) from different formulations of translation – for example, classical and axiomatic translations.
- Comparison of final results of resolution (Proof/Completion found) from the translation of a set of target formulae, to the same results from a prototype implementation of the axiomatic translation in Prolog [m12dfg as described in 1].

The software testing process listed above is a time consuming process. The expected results for the last three sets of tests can be seen in the results section figures (fig. 7.5-7.8, 7.14, 7.17-7.18). The prototype software released on the project web site passes all the tests. It will become apparent that each mode of testing has it's own particular strengths, and taken together, they ensure the correct functioning of the prototype. Elements from each of test methods were used repeatedly throughout the software development process.

For *unit testing*, as any new function was added to the eml-module, print statements were added to show the input and output, and a variety of problems were run through extended-SPASS, each designed to exercise the new code in a different fashion. The output was examined manually, or by using simple shell scripts, to confirm that the code was performing as expected. Print statements were left in place as further related functions were added, so that the impact of these new functions could be assessed. When an area of code was fully completed, the print statements were commented out, and in the final production code they were removed. This method was found to be highly efficient in finding numerous errors in the code, and in solving problems with the details of the design of the code. It should be noted that the way in which the code was designed aided this unit test process; where possible new code was added in discrete functions with a small and well-defined purpose. When such code had been found to be reliable, there was no need to specifically re-test it when other functions were added. The code that passed unit testing was in the most part, error free.

Extended-SPASS produces copious output that is primarily intended to help the user understand the axiomatic translation process (described in section 5). This output can

also be examined to determine whether the software is performing a proper translation. In numerous test cases, this output was checked by *visual inspection*, and found to be as expected. This method of testing was quite successful in detecting errors. It is however important to realize that there is (necessarily) some separation of the code required to actually perform the translation of the target modal formula, and the code printing details to the user (that is the two tasks are performed by separate and distinct subsections of code). Hence, even though the printed output is correct, this does not guarantee that translated TERMS are correct. Several errors of this type were found, and when the bugs were tracked down they mostly related to problems with the code producing information for the user, rather than code performing the actual translation. Several errors in program flow did exist. Most were detected by examining the output, but in two cases they were hidden; the output produced was correct, but the TERMS produced were incorrect. These problems were detected using the other testing methods described below.

A duplicate translation system was developed in Java (Jasper). A subset of the problems listed in figure 7.4 were submitted to extended SPASS, and the FINAL TERM (that is passed to the SPASS resolution prover) was captured and compared, subterm-by-subterm, to the same translation performed (independently) by Jasper. The target formulae used were chosen for their unusual properties and are marked in figure 7.4. They were formulae which had highlighted errors in other testing procedures, or that produced counter-examples which are able to highlight the differences between various formulations of modal-axioms and axiom combinations (see for example figures 7.10, 7.11, 7.18). The test set covers all the modal-axioms available in extended-SPASS, and both uni-modal and multi-modal target formulae, with both compositional and non-compositional translations. In each case there was an *exact correspondence* between the subterms of the respective translations. There are two purposes for this system. First, it provides a baseline of expected results (across a wide range of problems) that can re-run whenever the code of extended-SPASS is modified. A full test cycle, with 5280 test cases, takes around 3-4 hours to complete (this is a considerably shorter time than re-running the entire test set that is tabulated in figure 7.7-7.8, 7.17-7.18), and the analysis of the results is simple – just `grep`-ing out (and failing to find) the text for an error condition from the logs or from the text printed to standard-output. An important property of this test procedure is that it *ensures that any new code added does not disturb the pre-existing functionality*. In addition, although the two code bases are intended to perform *exactly the same translation* (so that subterms can be compared), the design of the systems is very different (as is necessarily the case – extended SPASS is written in C, and Jasper is object-oriented Java) and so the details of the methodology used to perform the translation is very different.

Because of these differences, comparison of the output is likely to highlight errors in logic and program flow.

During testing the following errors were found. Several typographical errors in the extended-SPASS code-base were detected, where code had been commented for unit testing, and then not uncommented later. The subterm-by-subterm comparison was critical to success in these cases. In addition, two subtle errors that involved production of additional unnecessary terms in the translation (one specifically for axiom 4, and the other effecting all axioms in specific circumstances) were detected. The feature critical to success here was the exploitation of the differences in the design of the two software packages.

Comparison of the results of execution of extended-SPASS (Proof/Completion found) using classical and axiomatic translations is very useful for error checking. Hence, the outcomes from the translation of a particular target formula, with a particular axiom instantiated using the axiomatic schema, should be the same as the result when the same axiom is defined by a correspondence property. Since these are independent translations, they can be crosschecked against each other. In the case of the ‘new’ axioms not described in [1] (see figure 7.15), this method of testing is particularly useful, because the axiomatic schema encodings were not taken from a published source, and is hence prone to possible error (although none has been identified). There are several difficulties with this approach. In particular, when the data available for the comparison is incomplete, that is, cases in which axiomatic translation succeeds, but insufficient time is allocated to produce a result from the classical translation (this time may of course be infinite). This difficulty arises because of the inefficiency of the classical translation, particularly for axioms $4/4^k$, $5/5^k$ and alt_1/alt_1^{kk} . (But, this was the reason that the axiomatic translation was developed). In addition, there are only two useful outcomes from the resolution prover, Proof and Completion found, and it would be expected that incorrect code would on occasion produce the correct result for the wrong reasons. This poor discrimination is addressed by performing calculations for a wide variety of target problems. Despite the difficulties, in *every case* where it was possible to make the comparison, the outcome of the classical translation of each target formula with each single axiom or axiom combination corresponded precisely to the outcome of the equivalent axiomatic translation. Hence, the outcomes are identical for axioms including KT vs. KT_c , KB^k vs. KB_c^k , KD vs. KD_c , $K5_o^k$ vs. $K5_c^k$, $K4^k$ vs. $K4_c^k$, $Kalt_1^{kk}$ vs. $Kalt_{1c}^{kk}$, $KT4$ vs. KT_c , $KT4$ vs. KT_{4c} , KTB vs. KT_{cB_c} , KDB vs. KD_{cB_c} , $KD4$ vs. KD_{4c} , $K4_oB$ vs. $K4_{cB_c}$, $K5_oB$ vs. $K5_{cB_c}$, $K5_oT$ vs. $K5_{cT_c}$, $KT4_oB$ vs. KT_{4cB_c} , $KD4_oB$ vs. KD_{4cB_c} etc. The same is true for mixed formulations of the axioms, $KT_{4c}B_c$ vs. $KT_{4c}B_c$, KD_{cB} vs. KD_{cB_c} , and KD_{c4} vs. KD_{c4_c} .

In the same fashion, crosschecking between different formulations of the same axiom combination is informative; it suggests target formulae that need to be checked more carefully for errors in the translation. Hence, logic S5 may be defined using the combinations $KT4B = KD4B = KT5 = KTB5 = KT4B5 = KT45 = KD4B5 = KDB5$. The difficulty here is that not all formulations appear to be complete (see section 7.2, for a discussion). Nevertheless, such comparisons did yield formulae that were checked manually in more detail for potential errors.

The final test series compared the results of execution of extended-SPASS (Proof/Completion found), over all the target formulae in the test set (indeed, target formulae 1-119 were developed from formulae present in the source code for `m12dfg`; additional formulae were added to test aspects of the translation of terms involving false (\perp) and true (\top)), to the outcomes from a prototype implementation of the axiomatic translation in Prolog [`m12dfg`, mentioned in 1], with the `.dfg` files produced by `m12dfg` executed in SPASS 3.0. There were minor problems with `m12dfg`, which were patched for this study. These bugs came to light during the comparison process. Briefly, in `m12dfg`, handling of reverse implication was broken, as was parsing of some axiom names involving the hat (\wedge) symbol. It is worth noting that for axioms 5 and 5^K , the syntax required for `mldfg` input is $(5_0)+(5)$, $(5^{\wedge}2_0)+(5^{\wedge}2)$, and $(5^{\wedge}3_0)+(5^{\wedge}3)$. The input and output files associated with the use of `m12dfg` in this study are available from the project web site (`m12dfg` itself is available from R. Schmidt). This was a useful test system. In particular it provided an initial confidence that the translation in extended-SPASS was correct, early in the development process. Again, the design of this translation system in Prolog is very different from the C-code of extended-SPASS, and comparisons are likely to find errors in program flow and logic, as well as specific problems with individual translation routines. Again, there are several problems with this test method. First, considering the way in which axiomatic translation works, if there are additional, unnecessary `TERMS` produced during the translation, then there may be no difference in the outcome of resolution. For example, if the translation requested was 5_0 and the actual translation delivered was 5_05_05 , there would be no difference in the outcome of resolution, except that the unnecessary terms might slow down the calculation (possibly yielding no solution). In this case, the translation performed would clearly be incorrect, but the error would be undetected. Indeed, a program logic error of just this type occurred, and was not picked up by this method of testing (although it was eliminated by other tests). It should also be noted that the time required to run the problems in both systems is several hundred hours, making this an impractical method for testing other than a completed system (or at milestones during the development process). Another problem with this test method is the

more limited range of axioms available in `m12dfg` (axioms K, T, D, B, 4, 5 alt_1 , alt_1^{κ} , 4^{κ} , 5^{κ} , M and B^{κ} , with only partial coverage for axiom D), making it impossible to crosscheck all the range of output for extended-SPASS. In addition, there is a problem with the efficiency of the translation produced by `m1dfg`; many subterms are unnecessarily duplicated, making automated comparison of the output TERMS from extended-SPASS and `m12dfg` impossible. It is also not possible to make a subterm-by-subterm comparison between extended-SPASS and `m12dfg`, because the format of the axiomatic schema encoding for axioms is seldom the same as used in extended-SPASS. For example, in `m12dfg` the axiom T is translated as $\forall x(Q_{\Box r,p}(x) \rightarrow Q_{r,p}(x))$, and the quantification is different in many cases, for example the axiom $\text{alt}_1^{2,2}$ is $(\forall x,u,v,y,z((R_r(x,u) \wedge R_r(u,y) \wedge Q_{\Box r,p}(y)) \rightarrow (R_r(x,v) \rightarrow (R_r(v,z) \rightarrow Q_{\Box r,p}(z))))$). Clearly these formulations are equivalent to those used in extended-SPASS (figure 2.8), but it is not easy for simple test software to recognize them as identical. Finally, the translation in extended-SPASS is much more efficient than in `m12dfg`, and so comparisons between data from the two implementations is made more difficult by the incomplete data from `m12dfg` (problems that are not solved). It is likely that the reduced efficiency is caused by the difference in formulations of the schema encodings, and by the many duplicate TERMS that are produced by `m12dfg`. Note that the translation in `m12dfg` is made ‘offline’, and so extended-SPASS has an additional burden of translation and checking for duplicate TERMS that does not appear to slow it down significantly.

7.2 Results.

This section describes results obtained using extended-SPASS to solve problems in modal logic. The method used was in many cases to access one of a series of preformulated `.dfg` files, applying a series of modal-axioms and axiom combinations, to the target formulae defined in the `.dfg` files, as is shown in figures 7.1-7.3. Note in particular the default options chosen in the `list_of_settings` section. The target modal formulae used are tabulated in figure 7.4. These are an extended set of formulae referred to in [1]. A full set of the template files and scripts used to generate results is available at the project website. In many cases data is analyzed for all the target formulae in 7.4. In other cases, the test formulae involving exploring just the handling of true and false (formulae 120-133) are excluded; sometimes formulae that took an excessive amount of time to calculate (formulae 91-104) are excluded; in other cases the multi-modal formulae (118-119) are excluded. In general the execution of the resolution process was

terminated after 200 seconds if no result has been found. This figure proved a good compromise between problems solved and excessive execution time.

Some features of the results (tabulated in the figures 7.5 onwards) are now discussed. In some cases the data quoted for outcome (Proof/Completion found) is a composite of two sets of experiments. Between these experiments a few new features and optimizations were added to extended-SPASS. It is important to note that the outcomes did not change between these experiments (that is Proof and Completion did not change), but in the case of calculations that were terminated early by the 200 second timeout, the second implementation may have been a little more efficient (since 23 more problems were solved in the second dataset that were not solved in the first; although 9 cases were solved in the first data set, but not in the second). The execution time data refers only to the second set of experiments. Where experiments were repeated within a single dataset, the typical error in measurement of execution times seems to have been around 10% of the total execution time (under normal load conditions). It is not certain what the origin of this difference may be, but the method by which SPASS measures execution time seems to be susceptible to interference by CPU load. In tests, where experiments were repeated in different more extreme load conditions, the execution times reported by SPASS varied with a standard deviation of up to 25% of execution time. Modification of code in SPASS responsible for measuring timing is outside the scope of this project. (Note that this finding has little impact for the timing data presented later. For most experiments, each SPASS process had exclusive access to the CPU/memory, and only general trend information is extracted from the timings, over a wide range of different experiments).

The outcomes from the SPASS resolution prover in extended-SPASS are tabulated in figures 7.5-7.8 and 7.17-7.18. These values were used as a reference set during testing in the development process (see section 7.1). This local satisfiability data is also interesting in its own right. Axioms and axiom combinations that were described in [1] did not have reference values quoted for the outcomes. These are now available in figure 7.5-7.8. In addition there are reference outcomes quoted for axioms not mentioned in [1] (figures 7.17-7.18).

It is worth remembering what the outcomes Proof found and Completion found mean in SPASS. If the classical semantic translation of a target formula T ($\Box p \rightarrow p$) is considered ($\forall x(\forall y(R(x,y) \rightarrow Q_p(y)) \rightarrow Q_p(x))$), and the correspondence property for axiom T is used ($\forall xR(x,x)$), then it is clear that formula T is valid in KT , and $\neg T$ is unsatisfiable in KT . Hence, in SPASS:

1. Axioms list:	$\forall xR(x,x)$
Conjectures list:	$\forall x(\forall y(R(x,y)\rightarrow Q_p(y))\rightarrow Q_p(x))$
Yields outcome:	Proof
2. Axioms list:	$\forall xR(x,x) \ \& \ \neg\forall x(\forall y(R(x,y)\rightarrow Q_p(y))\rightarrow Q_p(x))$
Conjectures list:	None
Yields outcome:	Proof
3. Axioms list:	$\forall xR(x,x) \ \& \ \forall x(\forall y(R(x,y)\rightarrow Q_p(y))\rightarrow Q_p(x))$
Conjectures list:	None
Yields outcome:	Completion

In a similar way, in extended SPASS the formula named ‘T’ ($\neg(\Box p\rightarrow p)$) yields Proof found in axiom KT, and Completion found in K (see table 7.5). In the tabulated results (as expected) most target formulae yield Completion in K (least restricted frames) and yield Proof in S5 formulations (most restrictive frames). There is a general trend for target formulae to be more likely to yield the result Proof as the restrictiveness of the applied modal-axiom combination increases. (See figures relating modal-axioms in [30] and [10] in order to gauge the relative restrictiveness of modal-axioms). It can be seen that the range of problems chosen for the test set is sensible. There are a variety of outcomes according to which modal axiom is applied, and the problems range from easy to difficult when the resolution calculation is restricted to 200 seconds maximum execution time.

The results for the execution times, and the numbers of formulae that were solved (that is, gave a result Proof or Completion found) before the timeout was reached (successful) are seen in figures 7.9 and 7.13. First it is worth noting the total execution times for the axiomatic translations using schema encodings of axioms (most of figures 7.5-7.6) is 22.4 hours, whilst the same data for the classical correspondence properties (figures 7.7-7.8) is 92.1 hours. This indicates that the axiomatic translation tends to be faster. Likewise, the total number of problems solved for these data are 2360 and 1167 respectively. Thus the axiomatic translation is more often successful than the classical translation. In general, as the number of box or diamond operators in the target formula increases, the size of the instantiation set increases, the problem is more difficult to solve, and the target formula is less likely to be solved before the execution time cutoff, or takes longer to solve. Examining the data for the individual axioms (figure 7.9), it appears that the best indicator of the effectiveness of the axiomatic translation is the number of successful calculations (that produce a Proof or Completion result). Where this figure, for any particular axiom or axiom combination, is similar for both the axiomatic and classical translations, then it is informative to examine the execution time statistics in order to judge the effectiveness of the axiomatic translation. It is seen that:

- Problems in axioms T, B, and D are easily solved in both classical and axiomatic translations. For axioms T and D, the execution times are smaller for the classical

translation, so the axiomatic translation is slower. For axiom B, the two translations have similar properties.

The correspondence properties for T, B, and D belong to fragments that are decidable by ordered resolution (like the clauses produced by the axiomatic translation). It seems likely that the large numbers of new TERMS that are created by the axiomatic translation, and are subsequently passed to the resolution prover, are largely responsible for the slower execution times in those problems that are easily solved by using the correspondence properties. The additional overhead incurred by the axiomatic translation itself, in processing the schema encoding, may also contribute to this slower execution. It is worth noting that the axiomatic translation is intended to aid in the solution of more difficult problems than these, and that for these easy problems the slowdown is fairly small, so that the axiomatic translation never imposes a crippling overhead on the calculation.

- As expected, for axioms 4, 5, and alt_1 , the axiomatic translation is significantly better. More problems are solved, more quickly. Only 21%, 40%, and 15% of target problems are solved using the correspondence properties for axioms 4, 5, and alt_1 respectively. In contrast, 100% of problems are solved using the schema encoding of these axioms.

It is expected that the axiomatic translation will be much better for these modal-axioms. The transitive (4), Euclidian (5), and functional (alt_1) properties, as defined by correspondence properties, are difficult for resolution provers to solve in the classical translation [1, 39]. The clausal forms have no single maximal literal, and as a result a great (perhaps unbounded) number of inferences may be required for resolution when the problem is defined by the classical translation. In the axiomatic translation, the target problem is reduced to fragments that are all decidable by ordered resolution (true not only for these axioms, but for all translations [1]). The major contribution of the axiomatic translation is that it provides a decidable (by ordered resolution) translation for target problems defined in these axioms (4, 5, alt_1).

- This same trend applies even more clearly to the higher order variants of axioms 4^k , 5^k , and $\text{alt}_1^{k1,k2}$. The target formulae in the modal-axioms 4_c , 4_c^2 , and 4_c^3 are increasingly difficult, with fewer problems solved (in that order), as are $5_c < 5_c^2 < 5_c^3$, and $\text{alt}_1 < \text{alt}_1^{1,1} < \text{alt}_1^{2,1} < \text{alt}_1^{1,2} < \text{alt}_1^{2,2}$. The axiomatic translation is better than the classical translation in every case, with more problems being solved, more quickly. However, the higher order axioms do not perform as well as ‘base’ axioms (4, 5, and alt_1) in the axiomatic translation; significantly less than 100% of problems are solved.

These higher order axioms will benefit from the same improvement as the base axioms (described above), but they also contain chains of TERMS in the schema encodings (of the

type $R(x,y) \wedge R(y,z) \wedge R(z,v) \wedge \dots$). These are likely to cause difficulties for the resolution calculation of the axiomatic translation, with a large number of inferences arising from these chains.

Examining the data for the combinations of axioms (figure 7.9 and 7.13), several features can be seen:

- The axiom combinations, problems in TB and in DB are easily solved in both the axiomatic and classical translations. In both cases, the resolution calculation is faster in the classical translation, than in the axiomatic translation.

It should be noted that axioms T, B, and D are all easily solved for calculations on the target problems with a single modal-axiom applied, and hence the axiom combinations TB and DB are both made exclusively from components that are themselves easily solved. Again, for the same reasons as before, it is not too surprising that problems solved easily in the classical translation are slower in the axiomatic translation. (Note also that D is valid in T, and hence the combination DT would be equivalent to T; this is the third possible combination of these axioms).

- For the other axiom combinations, 4B, 5B, D4B, T5 and T4B each have somewhat less problems solved, and are solved somewhat slower in the classical than in the axiomatic translation. Axioms combinations D4 and T4 each have substantially less problems solved, and are solved substantially slower in the classical than in the axiomatic translation.

It appears that for more difficult problems, involving combinations of the axioms including the (more difficult) axioms 4 and 5, and then the axiomatic translation is much superior to the classical translation. It is a little subjective (because different measures can be used) but approximately the ‘difficulty’ of calculations, and also the degree of improvement offered by the axiomatic translation is ordered $TB < DB < 5B < 4B < 5T < T4B < D4B < D4 < T4$, with the more difficult axiom combinations, which show a large improvement in the axiomatic translation, on the right, and the easy problems that are more readily solved in the classical translation on the left.

- Comparing the mixed classical and axiomatic translations (T_c4B_c vs. $T4_oB$. D_cB vs. DB , and D_c4 vs. $D4$), all problems are solved in each case, but looking at the execution times, in each case the mixed translation is significantly faster than the axiomatic translation.

The axiom combinations expressed as mixed translations were presumably chosen in [1] because many of the component axioms are easily solved (T, B, D), and have lower execution times in the classical than in the axiomatic translation. In each case, the axiom

that is incorporated as a correspondence property is either axiom D_c , B_c or T_c , and based upon the good performance of each axiom in the correspondence property, it might well have been expected that the mixed translation would be superior. An attempt is being made to choose axiom formulations in which the best features of the axiomatic and classical translations are combined. It is also interesting to make a comparison with the purely classical translations. For D4 and for T4B the mixed translation is better than the axiomatic translation, but these are both significantly better than the classical translation. For DB the mixed and classical translations are similar, and both better than the axiomatic translation.

Based upon these results, it would be worth looking at mixed translations for the axiom combinations T_c4 , D_c4B_c , $4B_c$, $5B_c$ and T_c5 , which may show better properties than the purely axiomatic translation. It will have been appreciated that the improvements achieved using the mixed translation may sometimes be quite modest. However for large problems (larger than the target formulae in the test set) such optimizations may represent a significant improvement, and so will be worth investigating. Proof of complete and sound translations will be required before such optimized mixed forms of translation can be used.

There are important differences between the execution times for target formulae yielding either Proof (which are unsatisfiable) or Completion (which are satisfiable) as outcomes. In general, target formulae with the outcome Proof are solved more easily than target formulae with the outcome Completion. The mean execution time for problems yielding Completion is more than 4.5 times greater than for those yielding Proof. It is clear that for unsatisfiable problems, the search space will tend to be smaller, since the calculation will be terminated when the empty clause is derived. For satisfiable problems, the calculation only terminates when a saturated set of clauses is derived. The difference between satisfiable and unsatisfiable problems is most dramatic for the axioms 4^k_c , 5^k_c , $alt_{1c}^{k1,k2}$, T_c4_c , 4_cB_c , 5_cB_c , T_c5_c , $T_c4_cB_c$ and $D_c4_cB_c$. In these cases no satisfiable problems are solved in the classical translation (see the large blank section in the table 7.9). These difficulties are however solved in the axiomatic translation (which incidentally is how the problems have been identified as satisfiable). *The improvements for the axiomatic translation are most substantial for satisfiable problems* (with outcome Completion). For problems in the easier axioms (T, B, D, TB, DB) the differences observed are much less pronounced. For problems solved with the result Proof, table 7.13 shows that the greatest improvements in execution time for the axiomatic, versus the classical, translation are seen for axioms 5^2 , 5^0 , $alt_1^{1,1}$, $alt_1^{2,1}$ and D4. In some of the other cases, the small amount of data available, or missing data, restricts the usefulness of any interpretation that can be made.

Tables 7.10 and 7.11 give a summary of the counter-examples that discriminate between non-complete and potentially complete formulations of various axiom combinations. After reading the text for [1] it was *expected* that, in any combination of axioms involving the modal-axioms B, D, T, $4/4^k$, $\text{alt}_1/\text{alt}_1^{k1,k2}$, that complete axiomatic translations would be created simply by incorporating each axiom into the translation with relevant compositional terms (see table 2.18) added to all the subsequent axiom(s) in the particular sequence. So, for example, the rather arbitrary combination $4_o.\text{alt}_1^{2,2}.T$ might be expected to be complete. (In some cases compositional terms might be omitted, but this would be an optimization that needed to be proven). Only for axiom $5/5^k$, it appeared, was there a complication that the compositional terms needed to be applied for the axiom itself ($5/5^k$), and not just to the subsequent axiom in a sequence. While this ‘general result’ was not stated in [1], neither was it made clear that this would not be the case. As a result of this apparent ‘general case’ it appeared that axiom combinations, like those making up S5, offered a very useful method of testing the implementation of extended-SPASS: Different well-known equivalent axiom combinations (like S5; see tables 7.10 and 7.11) would be executed for the target problems, and in each case they should yield the same outcome. It seems however that formulations following this simple pattern are *not necessarily complete*. (The expected outcome for a complete translation is judged using a purely classical translation, or an axiomatic translation in a formulation that had been shown to be complete in [1]). The difficulty is that the instantiation set required for such problems (see for example section 2.2.3) is not as simple to define as expected. Needless to say, thoughts of using these equivalent formulations of axioms as a test mechanism were abandoned.

These results point to further work that needs to be undertaken in defining the principles of the axiomatic translation, and in particular further theoretical proofs that need to be undertaken. There was no time during this project to investigate these proofs (and it was not part of the project specification), but these results do definitely demonstrate the utility of extended-SPASS in investigating the properties of the axiomatic translation. It will be interesting for further work to define the minimum instantiation set that is needed to produce a complete translation. The minimum instantiation set is of interest, since it yields a smaller number of translated sub-formulae that are likely to be solved in a shorter time, with a smaller proof. The results in table 7.10 and 7.11 suggest formulations that are worth investigating, since they are likely to be complete.

In the next chapter, extended-SPASS is used to screen counter-examples for logics like S5, for potentially complete formulations of axioms combinations. When only these counter-examples are screened, rather than the full test set, the execution times for experiments are substantially decreased. Two hypotheses are tested. First that complete

formulations are created if all axioms (B, D, T, $4/4^k$, $alt_1/alt_1^{k1,k2}$) are formulated for composition in the same way as axioms $5/5^k$; with compositional terms applied at both the axiom itself, as well as at subsequent axioms in the sequence. This is achieved with axioms defined in the format (for example 4_04 , D_0D , etc). The second hypothesis tested is that the complete formulation for axiom combinations may be built in the following way; if a combination of axioms is $X \equiv X_1X_2..X_n$ (where X_i is one of B, D, T, 4^k , 5^k , $alt_1^{k1,k2}$), then a complete translation is produced by a translation of the format $X_oX \equiv ((X_1)_o(X_2)_o..(X_n)_o)X_1X_2..X_n$. Both these cases manipulate the sub-formulae added to the current instantiation set, although in different ways. See section 8.2.1 for the findings. It is clear that features such as those described above need to be addressed before the implementation of an automated translation for an arbitrary modal axiom is implemented.

The table 7.11, in which the execution times of incomplete and potentially complete formulations of the modal-logic S5 are compared, illustrates some commonsense properties of the axiomatic translation. The trends listed below are followed in general, but there are significant exceptions in most cases. It is likely that the difficulty of solving a particular clause set that arises from the translation of a particular target formula and axiom combination, is an important property, and unfortunately it cannot be taken into account by when extracting trend information.

- It would be expected that formulations with more component axioms would tend to have a slower execution time in resolution. Thus D45B, and T45B would be the slowest (in practice, true for at least D45B), and that T5 would yield solutions that tend to be faster. This is because D45B and T45B will introduce additional sub-formulae into the translation.
- The presence of axioms that are easy to solve (that is, T, D, and B) among the components of the formulation of S5, should reduce execution time. This is difficult demonstrate, since every formulation contains one or more of these axioms.
- In the same way, formulations containing axiom 5 might be expected to take longer to execute than formulations with axiom 4, since calculations for target formulae in these axioms alone show axiom 5 to be more difficult to solve than axiom 4. All the formulations of S5 investigated contain either axiom 4, or 5, or both. This trend is broadly upheld.
- In cases where additional (unnecessary) axioms are included in the formulation, the additional sub-formulae introduced might be expected to slow the calculation. Hence, in general, solutions in 5T are faster than in 5TB, in D4B are faster than in D4B5, and in T4B are faster than T4B5.

- Formulations that do not use composition (whether complete or not) should be faster to solve than formulations in which additional compositional sub-formulae are introduced. This trend is upheld so that, for example, solutions in T4B are faster than $T4_0B$, TB_04 , etc, and in D4B are faster than in D_04B , etc.
- S5 is a restrictive logic, and so target formulae are difficult to solve in S5. Hence, it would be expected that any classical translation would be significantly slower than the axiomatic translation. This trend is definitely upheld; all the classical translations are at the bottom of the table (table 7.11). Note, of course that the order in which axioms appear in the classical translation is of no consequence.

Table 7.12 shows the execution times for the eml-module (for the eml-translation to first-order logic). It demonstrates that the implementation of the axiomatic translation is sufficiently fast to not cause any significant reduction in the overall rate at which problems are solved, especially for the more difficult problems where the execution time is expected to come close to 200 seconds (the cutoff time used). In such cases, then less than a second for the translation is insignificant. It is known that the axiomatic translation itself is executed in linear time [1]. In the next chapter, the complexity of the schema encoding in the axiomatic translation is briefly investigated. In the current implementation, one optimization in particular can extend the execution time; the repeated checking of newly added TERMS to ensure that no duplication exists. In tests, there is little execution time added by this duplicate checking. A large target formula was developed with all the subformulae repeated 100 times. Less than 8% was added to the execution time for the translation component. It is important to appreciate two related points. The axiomatic translation is in general very quick as compared to the overall execution time, because resolution is in general a very slow process. It has already been suggested that the extra overheads incurred by checking for duplicates are small compared with the time required to handle duplicates in the resolution prover; hence the implementation `m12dfg` is slower than extended-SPASS, despite performing its translation offline in a pre-processing stage, probably largely because it passes duplicate TERMS to the resolution prover (see section 7.1).

Table 7.14 shows some results from the implementation of the Scott-Lemmon translation of correspondence properties from modal axioms. Using this syntax, it is clear that a great many modal axioms can be defined in correspondence properties without needing to modify the code of extended-SPASS. This is a step towards the generation of an automated axiomatic translation for an arbitrary modal axiom, since it provides a baseline against which a translation of the same arbitrary modal axiom can be tested.

Several axiomatic schema encodings for modal axioms that were not discussed in [1] are *proposed* in figure 7.15. These axioms were chosen after a review of some general texts on modal logic ([10, 26, 30]), and represent some of the many axioms that are commonly used in the literature. In each case a proof of the encoding is needed (but it was not part of the specification of this project to develop these proofs). Nevertheless, there is a high degree of confidence that these encoding will be complete. A general proof of the schema encoding for a large class of modal axioms would be useful. This would allow work on the development of an automated procedure to generate schema encodings to advance with confidence. (This is work for the future). Of particular note are the axioms McKinsey and Löb (axioms M and W), for which no first order correspondence property can be defined [1]. Although no specific proof is offered, the axiomatic translations for these two cases should still be sound and complete [1] (but, completeness needs to be proven). The derivation of the axiomatic schema encodings for all these axioms is given in figure 7.16. In each case, a particular formulation of the modal axiom has been chosen, and this gives rise to a particular schema encoding. In most cases, it is possible to generate several different encodings for each axiom, and it remains to be seen (in further work) what factors are important in optimizing the efficiency of the encoding in resolution. The particular encodings presented here try to follow the pattern (implicitly) presented in [1], with the smallest possible number of new sub-formulae being introduced by the encoding chosen.

Data for calculations on local satisfiability of the target formulae in these newly developed schema encodings are presented in tables 7.17-7.18. After the experience with incomplete formulations of S5 (described above), it was decided to take a safe course, and to implement these axioms in the same fashion as the encoding of axiom 5, with compositional TERMS applied to the axiom itself (as well as to subsequent axioms in any sequence that exists). This is because no proofs were offered for completeness. The evidence suggests that this approach was over cautious in the cases of axioms DBBB, DEN, G, SR and TR, since no counter-examples were seen in the outcomes for the local satisfiability calculations (for example, comparing G with G_0). Hence it is likely that these axioms could be formulated in the same fashion as axioms B, D, 4, and alt_1 . The execution times for such non-compositional formulations will be much faster (because there are less TERMS introduced into the translation). The new sub-formulae and compositional subformulae that are introduced by these axioms are listed in 7.15. Data for these axioms is presented because it can be crosschecked against outcomes from the correspondence properties (defined by the Scott-Lemmon syntax). In each case, all the outcomes were identical for these various formulations of the problem (except in trivial cases where no

result was produced before the cut-off time). A few experiments are presented for the axiom combination D.DEN. These investigate the use of composition with these newly developed encodings. The outcomes from all the formulations presented are the same. From the data, it seems likely that the combination D.DEN will be complete without compositional subformulae being included (again this requires proof in further work).

For axioms B^2 and B^3 , the implementation of the axiom encoding in the same fashion as axiom 5 seems necessary to ensure completeness. Compositional subformulae do need to be included in the instantiation set for translation of the axiom alone. Counter examples are listed in figure 7.18 for experiments in which the outcomes from axioms B^2 and B^2_o are different. (The potentially complete and incomplete formulation was identified by comparison to the classical translation). Data in axiom B^3 is less certain, because so few problems were solved in the classical translation.

The data presented for axioms M and W is separated from the other data in table 7.18 since it has not been as thoroughly tested, largely because of the impossibility of crosschecking the outcome of schema encoding with that for correspondence property. No counter-examples were identified among the target formulae (by comparing M with M_o and W with W_o), and so formulations of M and W may be complete, with the additional subformulae introduced in M_o and W_o being unnecessary. Significantly less target problems are solved when these compositional subformulae are included.

The test data generated for multi-modal axioms CR, CR2, CR3 is not presented. The output was checked manually and the expected translation was achieved (see section 2.2.7). The limited number of target formulae to which these axioms can sensibly be applied (CR and demri4) needs to be expanded (in future work) for a more thorough checking protocol.

A brief investigation of the execution time properties of the newly defined encodings for axioms DBBB, DEN, G, SR, TR, B^2 and B^3 is presented in figures 7.19 and 7.20. It was observed (data not presented) that the compositional formulation of these axioms takes much longer to execute than the non-compositional formulation (for example G versus G_o , etc...). Axioms DEN and SR are easily solved in the classical translation, and the classical translation is faster than the axiomatic translation. The efficiency of the two translations is similar for axioms TR_c and TR_o . For axioms $DBBB_o$, G_o , B^2_o and B^3_o , the target formulae are more difficult to solve, and in every case the axiomatic translation is more efficient than the classical translation. The improvements for axioms G_o , B^2_o and B^3_o are particularly good. Since the formulations DBBB, G, SR and TR may be complete, then the axiomatic translation might be expected to perform even better in these formulations. For axioms DBBB, G, TR B^2 , and B^3 , and to a lesser extent for axiom TR, the most

significant difference between the classical translations and axiomatic translation is seen when the outcome is Completion (see the large number of blank spaces in the tables). As before, the improvement seen for axiomatic translation is most significant when the outcome of resolution is Completion, so these are the problems that are not solved in the classical translation, but are solved in the axiomatic translation. This very large difference in execution time does not apply to the easily solved axioms DEN and SR.

The results for *global* satisfiability of the target formulae from the test set in a variety of axioms are shown in figures 7.21-7.25. In [1] it was stated that the same principles that applied to local satisfiability could be extended to global satisfiability, but no proofs were offered. The modifications to extended-SPASS that were necessary to support these calculations are discussed in the section 8 when the maintainability of extended-SPASS is considered. Similar patterns of results are seen in these calculations to those seen for the local satisfiability calculations. In particular it is worth noting that, again the axiomatic translation can be crosschecked against the classical translation, and that in each case, no counter-examples were seen between these formulations, providing evidence (but not proof) that the axiomatic translations for global satisfiability are complete. This holds for both single axioms, and for the axioms combinations that were examined.

Comparing the results of the local and global satisfiability calculations, the following points are seen:

- The data in table 7.21 took 2.4 hours to calculate, whereas the equivalent data in local satisfiability took 22.8 hours. Comparing the classical translation in table 7.23 with the equivalent data for local satisfiability, the execution times are 37.6 and 92.1 hours. It appears that global satisfiability is easier (and faster) to calculate.
- For axiom K, the global satisfiability calculations produce a mixture of Proof and Completion outcomes, as opposed to mainly Completion seen for local satisfiability. Proof is still the most likely outcome for target formulae in formulations of axiom S5.
- Axioms T_c , D_c , B_c , $T_c B_c$, and $D_c B_c$ are still easily solved in the classical translation. Probably because of the smaller overall execution times, the execution times in axiomatic translation and classical translation are very similar, although the axiomatic translation is still a little slower in some instances.
- The axiomatic translation is still significantly better than the classical translation for axioms $4/4^k$, $5/5^k$, $alt_1/alt_1^{k1,k2}$. It should be noted that the difficulty arising from chains of related subformulae in the higher order axioms 4^k , 5^k and $alt_1^{k1,k2}$ do not appear to be as significant in global satisfiability calculations. Hence for example, a similar number of solutions are generated for axiom 5 as for 5^2 , as for 5^3 . The execution times do however still increase in a regular fashion, so $4 < 4^2 < 4^3$, and $5 < 5^2 < 5^3$. Again, the

generally smaller execution times for global satisfiability will flatten the differences observed.

- A dramatic difference is seen when comparing calculations that have outcomes of Proof and Completion. In local satisfiability calculations, satisfiable problems (outcome Completion) are more difficult to solve, and show a very significant improvement when the axiomatic translation is used as opposed to the classical translation. This difference is reduced for global satisfiability. There are less blank areas in the tables. Only the axioms T_4c , T_5c , T_4cB_c , and D_4cB_c show no satisfiable problems solved. However, the trend discussed for local satisfiability is still present, and so satisfiable problem are still less readily solved than unsatisfiable problems across all but the easy axioms.

The capabilities of extended-SPASS have been demonstrated, and some of the properties of the axiomatic (and classical) translation of modal target formulae, in a variety of axioms and axiom combinations, have been investigated.