# Chapter 7: An Analysis of the Execution of FELIX

FELIX has proved many theorems. A few of these have been discussed in previous chapters. All of the proofs are in Appendix 3. This chapter presents data about the execution of FELIX on these theorems, and provides some analysis of this data.

## Speed of Execution

The data from the execution of FELIX in finding a proof of a theorem includes the number of tasks "generated" by FELIX in the course of finding that proof, with this number of tasks broken down in various categories. These categories include the number of tasks which were "processed", the number of tasks which were not processed, the number of tasks of each type (interest, adoption, all_detachment, exists_detachment, reductio), the number of processed adoption tasks used in the final proof, and the number of processed adoption tasks *not* used in the final proof. Also, the time taken to find the proof is recorded.

The execution time is for the implementation of FELIX in LPA MacProlog 3.5 on a Macintosh II with 8 megabytes of RAM.

In the following table, each problem has an ID (e.g. 5.1i) where the letter at the end of the ID indicates the logic mode in which the problem was executed. The '[F]' which follows an ID indicates that FELIX failed to find a proof. In each of these cases, this is the correct result (i.e. the problem as not a theorem in the logic in which that test was run). The times are given in seconds. When no number is given in a field of an entry, then that number is '0'. 'Ad' is "adoption", 'Int' is "interest", 'Ex' is "exists detachment", 'All' is "all detachment", and 'Red' is "reductio".

| | Time | Total Tasks | | | | | Unprocessed Tasks | | | | | Adoptions | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **ID** | **(Sec)** | *Ad* | *Int* | *Ex* | *All* | *Red* | *Ad* | *Int* | *Ex* | *All* | *Red* | *Proof* | *Nonproof* |
| 1c | 2.96 | 11 | 5 | | | 1 | | 2 | | | 1 | 8 | 3 |
| 1i | 2.8 | | | | | | | | | | | | |
| 2c | 0.51 | 4 | 1 | | | | 1 | | | | | 3 | 0 |
| 2i | 0.51 | 4 | 1 | | | | 1 | | | | | 3 | 0 |
| 3c | 16.98 | 46 | 21 | | | 4 | 9 | 8 | | | 2 | 11 | 25 |
| 3i | 2.63 | 8 | 7 | | | | 1 | 1 | | | | 6 | 0 |
| 4c | 2.88 | 8 | 6 | | | 3 | | 2 | | | 3 | 7 | 1 |
| 4i[F] | 1.35 | 1 | 5 | | | | | | | | | | |
| 5c | 7.71 | 20 | 13 | | | 7 | 2 | 1 | | | 7 | 12 | 6 |
| 5i | 6.28 | 20 | 13 | | | | 2 | 1 | | | | 12 | 6 |
| 5.1c | 4.41 | 14 | 8 | | | 3 | 2 | 3 | | | 3 | 9 | 3 |
| 5.1i | 6.96 | 18 | 17 | | | | | 3 | | | | 12 | 6 |
| 6c | 0.4 | 2 | 1 | | | | | | | | | 2 | 0 |
| 6i | 0.41 | 2 | 1 | | | | | | | | | 2 | 0 |
| 7c | 0.65 | 2 | 1 | | 1 | 1 | | | | | 1 | 2 | 0 |
| 7i[F] | 0.75 | 1 | 1 | | 1 | | | | | | | | |
| 8c | 7.06 | 16 | 2 | | 6 | 1 | 4 | | | 5 | 1 | 8 | 5 |
| 8i | 6.65 | 15 | 2 | | 6 | | 4 | | | 5 | | 8 | 5 |
| 9c | 1.6 | 5 | 1 | 1 | 1 | 1 | | | 1 | | 1 | 6 | 0 |
| 9i | 1.5 | 5 | 1 | 1 | 1 | | | | 1 | | | 6 | 0 |
| 10c | 25.16 | 27 | 1 | | 5 | 1 | 2 | 1 | | 2 | | 15 | 5 |
| 10i | 26.33 | 30 | 1 | | 5 | | 2 | | | 2 | | 15 | 5 |
| 11c | 12.65 | 20 | 13 | 2 | 3 | 7 | | | | 1 | 7 | 14 | 6 |
| 11i[F] | 10.98 | 9 | 21 | 2 | 2 | | | | | | | | |
| 11.1c | 13.61 | 23 | 13 | 2 | 3 | 7 | | | | 1 | 7 | 15 | 8 |
| 11.1i | 65.63 | 89 | 41 | 9 | 11 | | 11 | 3 | 7 | 6 | | 50 | 37 |
| 12c | 1.21 | 3 | 4 | | | 1 | | 2 | | | 1 | 3 | 0 |
| 12i[F] | 0.8 | | 3 | | | | | | | | | | |
| 15c | 1.41 | 5 | 2 | | | 1 | | | | | 1 | 5 | 0 |
| 15i | 1.3 | 5 | 2 | | | | | | | | | 5 | 0 |
| 15.1c | 3.33 | 12 | 5 | | | 1 | 1 | 1 | | | 1 | 9 | 4 |
| 15.1i | 3.2 | 12 | 5 | | | | 1 | 2 | | | | 9 | 4 |
| 16c | 3.81 | 11 | 5 | | 3 | 1 | 1 | 1 | | 2 | 1 | 7 | 3 |
| 16i | 3.65 | 10 | 5 | | 3 | | | 1 | | 2 | | 7 | 3 |
| 17c | 4.15 | 11 | 7 | 1 | | 1 | | 1 | 1 | | 1 | 10 | 2 |
| 17i | 4.03 | 11 | 7 | 1 | | | | 1 | 1 | | | 10 | 2 |
| 18c | 2.33 | 6 | 4 | | 2 | | | 2 | | 2 | | 6 | 1 |
| 18i | 2.35 | 6 | 4 | | 2 | | | 2 | | 2 | | 6 | 1 |
| 20c | 790.55 | 70 | 8 | 1 | 17 | 6 | 14 | 1 | 1 | 17 | 6 | 44 | 19 |
| 20i | 787.08 | 70 | 8 | 1 | 17 | | 14 | 1 | 1 | 17 | | 44 | 19 |
| 21c | 6.15 | 16 | 4 | | 6 | 1 | 1 | | | 6 | 1 | 11 | 4 |
| 26.1c | 79.78 | 80 | 4 | | 6 | 2 | 4 | | | 6 | 2 | 43 | 25 |
| 28.4c | 154.48 | 150 | 8 | 1 | 16 | 4 | 33 | | 1 | 16 | 4 | 57 | 58 |
| 29c | 2200.15[1] | 149 | 188 | | | 75 | 18 | | | | 74 | 36 | 83 |
| 29.1c | 174.76 | 86 | 99 | | | 37 | 16 | | | | 36 | 26 | 37 |

[1] This test was run with dilemma supposition disabled. With dilemma supposition enabled, the test found a proof in about 4500 seconds, but ran out of storage in preparing to print the proof.

Since infon logic is “weaker” than classical, one generally expects that the “same” theorem in the two logics will be easier to prove in classical logic than in infon logic. This is true of many of the problems listed in the table. However, there are some cases in which the infon logic mode proof of a theorem is found sooner than the classical logic mode proof; 3, 15.1, 16, and 18.

FELIX is not a fast theorem prover. There are many reasons why this is so. There are several aspects of its implementation which are slower than they could be: there are several places in which work is duplicated in the course of a search (because it was easier to implement it this way), and the Prolog system does not generate very fast code. Also, there are certain situations which are by their nature time consuming for the search algorithm. In problem 20 (the Schubert Steamroller), there is a great deal of time spent doing ‘all detachment’ processing. All detachment processing has a factorial complexity with respect to the number of “literals” in a disjunct developed from a universal formula and the number of formulae adopted in the current state of the search. Some subset of these literals may match the adoptions in many different ways (factorially many different ways, in the worst case). Since FELIX does a nearly exhaustive search among the possible matches, this can be extremely time consuming.

## Justifications used in proofs

The following table indicates which justifications are used in which proofs:

| *Justification* | *Problems* |
|---|---|
| all_detachment | 8c, 8i, 10c, 10i, 18c, 18i, 20c, 20i, 21c, 26.1c, 28.4c |
| bel_veridicality | 26.1c, 28.4c, 29.1c |
| bel_anti_veridicality | 29.1c |
| bel_negative_introspection | 29.1c |
| child | 21c, 26.1c, 28.4c, 29c, 29.1c |
| conditional | 1c, 3c, 4c, 5c, 5i, 5.1c, 5.1i, 11c, 11.1c, 11.1i, 12c, 15.1c, 15.1i, 16c, 16i, 17c, 17i, 18c, 18i, 21c |
| conjunction B | 4c, 5c, 5i |
| conjunction(1) | 3c, 3i, 11.1i, 26.1c |
| conjunction(2) | 11.1i, 26.1c |
| contrapositive | 2.1c, 5.1c |
| dilemma | 5c, 5i, 5.1i, 11.1i, 17c, 17i |

There are 40 distinct justifications listed in the above table. However, some of them are in commutative pairs: conjunction(1) and (2), disjunction(1) and (2), disjunction_to_conditional(1) and (2), equivalence(1) and (2), negated_disjunction(1) and (2). Counting each of these pairs as only one justification, there are 35 different justifications listed.

For the theorems proved, the justifications used in the most proofs are the conditional, existential_instantiation, and modus_ponens justifications.

These are the justifications which appear in the adoptions actually used in the proofs. Counts could also be made of the justifications used in non-proof adoptions and in unprocessed tasks.

## The length of the search

The number of interest tasks processed indicates the “length” of the backward AND/OR interest search and the number of adoption tasks processed indicates the “length” of the forward adoption search. The nonproof adoptions are those adoption tasks which were processed but which were not useful in reaching the goal state. There is no count made of the interest tasks which were processed but which did not play a role in the adoption of the goal formula. This would be interesting as a measure of how well focused the AND/OR search is. An estimate of this can be made by counting the “backward” justifications (those which have a ‘B’ suffix) in the final proof. This roughly corresponds to the interests “used” in the proof.