Andorra-I is an experimental parallel Prolog system which transparently exploits both dependent and-parallelism and or-parallelism. One of the main components of Andorra-I is its preprocessor. In order to obtain efficient execution of programs in Andorra-I, the preprocessor includes a compiler for Andorra-I. The compiler includes a determinacy analyser and a clause compiler, and generates code for a specialised abstract machine. In this paper we discuss the main issues in the Andorra-I compiler, presenting its abstract instruction set and describing the algorithms used in its implementation.

Abstract Andorra-I is an experimental parallel Prolog system which transparently exploits both dependent and-parallelism and or-parallelism. One of the main components of Andorra-I is its preprocessor. In order to obtain efficient execution of programs in Andorra-I, the preprocessor includes a compiler for Andorra-I. The compiler includes a determinacy analyser and a clause compiler, and generates code for a specialised abstract machine. In this paper we discuss the main issues in the Andorra-I compiler, presenting its abstract instruction set and describing the algorithms used in its implementation.

Keywords: PROLOG, Parallelism, Compilation Techniques.

§1 Introduction Andorra-I is a parallel logic programming system based on the Basic Andorra Model. It supports both dependent and-parallelism, by running determinate goals in parallel, and or-parallelism, by trying alternatives from non-determinate goals in parallel. Experience in using Andorra-I has led to the following main conclusions:

* Parts of this paper have appeared in “The Andorra-I Preprocessor: Supporting full Prolog on the Basic Andorra model”, presented at the ICLP91 Conference, and in “Performance of the Compiler-based Andorra-I System” presented at the ICLP93 Conference.
Andorra-I performs well and exploits parallelism successfully for Prolog applications, committed-choice style applications, and new, "Andorra style" applications.

The coroutining in the Basic Andorra Model can reduce the search space of logic programs. This holds true for some Prolog programs, and has been exploited in "Andorra style" applications.

The Andorra-I preprocessor \cite{15,12} is one of the key components of the Andorra-I system. The preprocessor was designed to obtain correct and efficient execution of Prolog programs in Andorra-I. To allow the correct execution of Prolog programs, we researched the operation of Prolog programs with traditional left-to-right selection function, and investigated which features allow early execution of goals, and for which features left-to-right execution needs to be enforced. To obtain efficient execution of programs we designed a compiler for Andorra-I. The compiler had to address the new characteristics of the Andorra selection function, and particularly the problem of determinacy detection. Therefore, the main innovation of the preprocessor over previous compile-time analysis tools was addressing two new problems: which goals can be executed early, and which goals must not be executed early.

We found that a goal must not be executed early if it interferes with the correct operation of builtins, such as a side-effect predicate or cut. This is rather hard to detect at run-time. We use the principle that some calls in the program are "sensitive", i.e., may behave incorrectly if goals to the right are executed early. We implemented a sequencer which at compile-time detects these calls, and generates code to restrict early execution of the goals that might interfere with them. The sequencer relies on mode information generated by an abstract interpreter. The mode information allows the sequencer to detect uses of cut and of meta-predicates that are not sensitive, thus preventing unnecessary sequencing. \cite{15}

To detect when a goal can be executed early, we need to know when it is determinate. The Andorra-I determinacy analyser generates for each procedure code to detect at run-time whether a call to that procedure is determinate. The determinacy code analyser was originally designed to compile only determinacy code, but was later extended into a full compiler for (Andorra-I) Prolog source programs. The extension is a clause compiler which generates abstract-machine instructions for individual clauses. The resulting code can then be run by the Andorra-I engine.

The language supported by the compiler is Andorra-I Prolog, which extends standard Prolog with the coroutining of the Basic Andorra Model. Andorra-I supports all the standard Prolog builtins, and can take advantage of pruning operators by using them to make goals determinate. The two pruning operators supported are cut, as in Prolog, and commit, as in Aurora. \cite{10}

In this paper we concentrate on how the determinacy code analyser and