Novel Type Systems for Concurrent Programming Languages
GR/L75177
Summary of Results

May 25, 2000

It has long been recognised that *statically typed* programming languages make the programmer's task easier by detecting many errors at compile time rather than leaving them to arise at run time. In such languages, the *types* (for example, *number* or *string*) of data items must be declared, and the compiler can check that operators are never applied to parameters of inappropriate types.

Concurrent programming languages introduce a new class of data item: the *communication channel*, which enables messages to be transmitted between components of a concurrent system. Clearly errors can result if the sender and receiver of a message do not agree on the nature of the message being transmitted, so it is beneficial to assign types to communication channels. The type of a communication channel specifies something about the data which can be sent along it: for example, that each message must be a number.

Much research has been done on type systems for concurrent programming languages. The particular strand which is relevant to this project has developed in the context of the *pi calculus*, a theoretical process algebra. One notable development in this area is the programming language Pict (developed by Pierce and Turner), a full-scale concurrent programming language which has at its core an implementation of the pi calculus and which incorporates a number of sophisticated typing features including *subtyping* and *higher-order polymorphism*.

In Pict, the type of a channel uniformly specifies that *all* the messages sent along it must have the same type. However, many communication protocols rely on complex sequences of messages of different types, and it is awkward to describe such protocols in a type system based on uniform channel types. The present project is based on a type system proposed by Honda *et al.*, in which the type of a channel can describe a sequence of messages of differing types: for example, first a number is sent in one direction, then a string is sent in the other direction, then this pattern repeats. Branching possibilities also exist, so that a choice made during one communication can affect the type of the subsequent dialogue. Such types are known as *session types*.

The main theoretical innovation of the present project has been the integration of a notion of *subtyping* into session types. It turns out that this enables backward-compatible enhancements to communication protocols to be described in a very natural way. The next step in the project is to implement session types, with subtyping, in Pict, to provide a realistic programming language in which the applications of this type system can be explored. After a further phase of theoretical work, polymorphism will be added to the present type system.

Work continues under grant GR/N39494.

Contact: Dr S Gay, Department of Computing Science, University of Glasgow, 17 Lilybank Gardens, Glasgow G12 8RZ.