Multicore processors are becoming increasingly prevalent, blurring the lines between traditional parallel programs, which use cooperative threading to reduce execution time, and interactive programs which use competitive threading to increase responsiveness. Many applications, from games to database systems, can benefit from a combination of the cooperative and competitive threading models. Unfortunately, languages and tools such as cost models designed for cooperatively threaded programs do not extend easily to features of competitive programs, such as thread priorities.

In this thesis, we develop a model that extends the cooperative paradigm to account for features of competitively threaded programs. The contributions of this model span several levels of abstraction. First, we include a cost model that extends existing models of cooperative threading in order to allow programmers to reason about the parallel running time and the responsiveness of interactive parallel applications. Second, we propose a language that neatly combines abstractions for both forms of threading, and enables reasoning about efficiency and responsiveness at the level of the source code. Third, we develop a scheduling algorithm that efficiently handles threads with both responsiveness and throughput requirements. Finally, we implement the language as part of a compiler for Standard ML, and evaluate it on a benchmark suite including a number of realistic interactive parallel applications.

Thesis Committee:
Umut Acar, Chair
Guy Blelloch
Mor Harchol-Balter
Robert Harper
John Reppy, University of Chicago
Vijay Saraswat, Goldman Sachs