IMPLEMENTATION OF TEMPORAL LOGIC PROGRAMMING
LANGUAGE Tokio

S. Kono, T. Aoyagi, M. Fujita*, H. Tanaka

Department of Electronic Engineering, University of Tokyo
* Currently assigned to Fujitsu laboratories

ABSTRACT

The temporal logic programming language, "Tokio" can be executed by a resolution of Interval Temporal Logic. The resolution consists of three parts, which are: the unification of the temporal variable, reduction including temporal operator, and interval control. The implementation of Tokio includes automatic interval length determination and stream-like temporal variable representation. At the end of this report, an abbreviated version of a Tokio interpreter written in Prolog will be shown.

1. TEMPORAL LOGIC PROGRAMMING LANGUAGE Tokio

Tokio is a concurrent logic programming language designed for hardware description, based on first order linear time temporal logic (LTTL) (Wolper 1981) and first order local interval time temporal logic (ITL) (Moszkowski 1983). Since ITL fully embraces first-order predicate logic, Tokio includes the capabilities of Prolog.

When compared to Prolog's unification and reduction processing, Tokio processing consists of the following three elements:

-- Unification of temporal logic variables that possess different values at different times.
-- Ordinary reduction and future reduction
-- Division of time intervals

The following chapters summarizes ITL used as an extension of LTTL, and discuss the methods of implementing the above three elements.

2. LOGIC OF Tokio

Tokio executes Local ITL on the basis of LTTL. This chapter describes ITL and Tokio logic. In non local ITL, the value of a variable can be determined for time intervals. On the other hand in the Local ITL the value of a variable can be determined for time axis.
Local indicates that the value of variables is only determined at the beginning of an interval, and does not depend on the final time of the interval. In this sense, LTTL and local ITL are equivalent. But ordering descriptions is easier in ITL than in LTTL because times are treated as intervals. The use of LTTL is also convenient, if it includes an automatic synthesis system and a verification system for logical circuits (Fujita 1983). The language to be proposed, therefore, must permit ITL descriptions by expanding LTTL's capabilities.

In Tokio, the LTTL operator @ (next) is a basic temporal operator. @p means that p is true at the next time, that is LTTL has a discrete time concept, as does Tokio. In ITL's view, @p creates a new interval and p is true in this interval. The other important operator is "&&" chop of ITL. p && q means that p is true in some interval and q is true in the succeeding interval.

To execute chop by a next operator, Tokio uses two variables. One is for the fin time of time interval, and the other for the indicator of interval terminating. Tokio propositions are generally determined for time axis, and variable values are associated with the interval. For an atomic predicate, except for the few temporal operators, the truth value of the predicate is dependent only on the time. Previous "@" operator sets the later variable to "not empty". That is, there must be a next time. The "next" operator of this type is called a "strong next". There is a "weak next" (wnext), which does not set the variable of interval terminating. In the weak next, if there is no next time that is the end of the interval, the whole formulae is true.

The Tokio program is a kind of Horn clause. The primitive temporal operators such as next or chop are not allowed in head of the Horn clause. This is a useful subset of first order theory. Using these Horn clauses, the other LTTL operators are defined easily. For example, using weak next the operator "#", "always" is defined as follows. This operator corresponds to the square of LTTL.

#P :- P,wnext(#P).

The syntax of the Horn clause is that of C-Prolog (Pereira 1984) In the following sections we discuss the unification and reduction of Horn clauses of ITL.

3. UNIFICATIONS FOR TEMPORAL VARIABLES

For a first-order predicate, the truth value is always determined from the meaning of its argument for all times. Accordingly, unification in Tokio is executed at all times. Operator "=" is only used to fetch the current value of a variable. The "@" (next) function is used to fetch a future variable value*. To avoid a circularly structured

* This next function should be distinguished from the next operator, even though we use the same symbol. The next operator is prefixed to a predicate, while the next function is prefixed to a variable.