Title	状態空間探索におけるキャッシングに関する研究
Author(s)	小池, 憲史
Citation	
Issue Date	1999-09
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/1318
Rights	
Description	Supervisor:平石 邦彦, 情報科学研究科, 修士

On Efficient Cache Management in State Space Search

Satoshi Koike

School of Information Science,
Japan Advanced Institute of Science and Technology

August 13, 1999

Keywords: concurrency, state explosion, sleep sets, state-space caching.

Concurrent processes consist of several sequential subprocesses each of which acts concurrently in cooperation with other subprocesses. The concept of concurrent process is an abstraction of programs on parallel computers and network protocols on computer networks. Therefore, the consideration of concurrent process is very important in developing these systems.

In developing concurrent processes, the following problems frequently occur, which are not considered in a single process.

Dead lock A situation that a concurrent process cannot continue its execution any more.

Live lock A situation that a concurrent process repeats its execution infinitely without an effective evolution for its purposes.

Inadequate termination a situation that a concurrent process terminates without an achievement of its purposes.

It is a hard task to find and solve these problems during the development of concurrent programs. This makes their development more difficult than that of programs with a single process. For this reason, various methods for developing concurrent program have been studied.

State-space searching is one of supporting methods for developing concurrent programs. It generates the state space of concurrent processes by executing each transition one by one, and searches the state-space for finding problems.

In searching the state space, it stores all visited states in a memory to avoid *double works*, a situation that some states are explored more than once. When creating a new state, it is compared with stored states to check that it is already visited. However, it is hard to store all visited states because of largeness of the state space. To overcome this problem, several methods to reduce the number of states to be explored are proposed. *Sleep sets* is one of these methods. It uses the following fact. Considering a sequence of transitions, we exchange the position of a transition with one of its neighbors when it is independent, and obtain a sequence which is also executable, and gives the same result as the original one.

Empirically, it is known that the double work can be avoided sufficiently by storing a part of visited states. State-space caching uses this fact. It stores a part of visited states on high-speed main memory, called *cache*. When the cache overflows, one of states in the cache is selected, and is deleted. Therefore, it is desirable that states which will be visited repeatedly have priority in this selection. By this reason, a decision in selecting states from the cache is very important. Godefroid and Holzmann proposed several heuristics for the selection, which are discovered through experiments. But they did not study theoretically.

In this research, we study the mechanism of an occurrence of double works, and obtain the following results.

> A double work occurs when a sequence translated by the following rules gives a state that is already visited.
>
> **Rule1** If one of neighbors of a transition is independent, then they can be exchanged.
>
> **Rule2** If two sequences have the same start and end state in all engaging processes, then they can be exchanged.

We propose a new algorithm for cache management and prove its correctness. The proposed algorithm stores states whose local states are in one of the following.

- A joining local state
- The successor of a branching local state

Where joining local states and branching local states are defined as follows.

Joining local state a local state whose input degree is greater than one.

Branching local state a local state whose output degree is greater than one.