Lazy evaluation is a computational scheme which delays the evaluation of an expression until its value is needed, trying to improve the performance particularly when dealing with large data structure. We apply this mechanism to a multi-context algebraic reasoning system which, on a large data structure called the nodes, efficiently simulates parallel processes each executing an algebraic reasoning procedure under a particular context (or a premise). Exploiting the lazy evaluation, we present an efficient implementation of MKB (multi-context completion procedure), called lz-mkb, and implement it in a functional, object-oriented programming language Scala which features the lazy evaluation mechanism.

MKB simulates closely related inferences made in different processes all in a single operation.

simulate COMPOSE and SIMPLIFY by REWRITE_1

\[
\begin{align*}
P_1 & \quad P_2 & \quad P_3 \\
E & \ a=b & E & \ a=b & E \\
R & \ c \rightarrow b & R & \ c \rightarrow b & \text{SIMPLIFY} \\
R & \ b \rightarrow c & R & \ b \rightarrow c & \text{COMPOSE} \\
\end{align*}
\]

\[
N \quad < a : b, \{3\}, \{\}, \{1,2\}> \\
\quad < b : c, \{2,3\}, \{1\}, \{\} >
\]

\[
N' \quad < a : c, \{3\}, \{\}, \{2\} > \\
\quad \text{move} \ \{3\} \text{n}\{2,3\} \\
\quad < a : b, \{3\}, \{\}, \{1,2\}> \\
\quad < b : c, \{2,3\}, \{1\}, \{\} >
\]

The experiments show that lz-mkb is more efficient than the original MKB implementation of Kurihara and Kondo in all the problems examined.