공유메모리 프로그램의 최초경합 탐지를 위한 접근역사 분석

(Analyzing Access Histories for Detecting First Races in Shared-memory Programs)

강 문 혜 † 김 영 주 †† 전 용 기 †††
(Moon-Hye Kang) (Young-Joo Kim) (Yong-Kee Jun)

요 약 공유메모리 병렬프로그램의 디버깅을 위해서 비결정적인 수행결과를 초래하는 경합을 탐지하는 것은 중요하다. 특히, 프로그램 수행에서 가장 먼저 발생하는 최초경합은 이후에 발생하는 경합에 영향을 줄 수 있으므로 반드시 탐지되어야 한다. 이러한 최초경합을 탐지하기 위해 최초경합에 참여할 가능성이 있는 후보사건들을 수행 중에 수집하는 기존의 기법은 접근사건들을 감시하여 후보사건들을 수집하고, 그들간의 병행성 관계만을 검사하여 경합을 보고한다. 그러나 이렇게 보고된 경합은 경합들간의 영향관계가 고려되지 않으므로 최초경합임을 보장하지 못한다. 본 논문에서는 내포병렬성을 가진 병렬프로그램의 수행 중에 수집된 후보사건들을 프로그램 수행 후에 각 내포수준에서 분석하여 영향받지 않은 경합만을 보고하는 기법을 제안한다. 제안된 기법은 임의의 내포수준까지 분석하여 보고된 최초경합이 그 내포수준까지는 영향받지 않은 경합임을 보장하므로, 상위 내포수준에 대한 재분석이 필요없는 효율적인 최초경합 탐지기법이다. 본 기법은 내포병렬성에서 후보사건들만 수집되면 최초경합을 탐지할 수 있으므로 기존의 기법에 비해서 현실적이고 효과적인 디버깅을 가능하게 한다.

키워드 : 공유메모리 병렬프로그램, 최초경합, 후보사건, 내포병렬성

Abstract Detecting races is important for debugging shared-memory parallel programs, because races result in unintended nondeterministic executions of the programs. Particularly, the first races to occur in an execution of a program must be detected because they can potentially affect other races that occur later. Previous on-the-fly techniques that detect such first races based on candidate events that are likely to participate in the first races monitor access events in order to collect the candidate events during a program execution, and try to report the races only from determining the concurrency relationships of the candidates. Such races reported in this way, however, are not guaranteed to be first races, because they are not determined by taking into account how they are affected with each other. This paper presents a new post-mortem technique that analyzes, on each nesting level, candidate events collected from an execution of a shared-memory program with nested parallelism in order to report only first races. This technique is efficient, because it guarantees that first races reported by analyzing a nesting level are the races that occur first at the level, and does not require more analyses to the higher nesting levels than the current level. The proposed technique facilitates more practical and effective debugging than the previous techniques, because it guarantees to detect first races if candidate events are collected from an execution instance of the program with nested parallelism.

Key words : shared-memory parallel programs, first races, candidate events, nested parallelism

† 비 회 원 : 경상대학교 컴퓨터과학과
turtle@race.gsnu.ac.kr
†† 학생회원 : 경상대학교 컴퓨터과학과
akates@race.genu.ac.kr
††† 종신회원 : 경상대학교 컴퓨터과학과 교수
jun@nongae.gsnu.ac.kr
논문접수 : 2003년 5월 7일
심사완료 : 2003년 10월 6일

1. 서 론

공유메모리 병렬프로그램에서의 가장 심각한 오류의 형태인 경합(race)[1]은 병행적으로 수행되는 스레드들이 하나의 공유변수에 대해 적절한 동기화 없이 적어도 하나 이상의 쓰기 사건으로 접근할 때 나타난다. 이러한

경합은 프로그램 상에 의도하지 않은 비결정적인 수행 결과를 초래하므로 반드시 탐지되어야 한다. 특히 병렬 프로그램에서 발생할 수 있는 경합 중에서 가장 먼저 발생하고, 다른 경합으로부터 영향을 받지 않은 최초경합(first race)[1-4]을 탐지하는 것은 최초경합으로부터 영향받은 다른 경합들이 사라질 수 있거나 새로이 발생할 수 있기 때문에 더욱 중요하다.

공유메모리 병렬프로그램의 최초경합을 수행 중에 탐지하기 위해서 최초경합의 가능성이 있는 후보사건들을 수집하는 기존의 기법[3,5]은 수행 중에 모든 접근사건들을 감시하여 후보사건들만으로 공유 자료구조인 접근역사(access history)를 구성하고, 수집된 후보사건들 간의 병행성 관계를 수행 후에 검사하여 최초경합을 보고한다. 이러한 기법에는 대상 프로그램의 유형에 따라 내포병렬성을 가진 프로그램에서 한 번의 수행으로 후보사건을 수집하여 최초경합을 보고하는 기법[3]과 순서적 동기화를 가진 프로그램에서 두 번의 수행으로 후보사건을 수집하여 최초경합을 탐지하는 기법[5] 등이 있다.

이렇게 보고된 경합들은 후보사건들간의 병행성 관계만을 검사하여 보고된 것이므로, 그 경합들이 최초경합임을 보장하지 못한다. 왜냐하면 각 내포수준에서 상위 내포수준의 후보사건들과의 영향관계를 고려하지 않고 현재 내포수준에서의 최초경합 가능성 여부만을 검사하여 후보사건을 수집하므로, 그들간의 병행성 관계만을 검사하여 보고된 경합 중에는 상위내포수준에서 영향받은 경합들이 존재할 수 있기 때문이다.

본 논문에서는 내포병렬성을 가진 공유메모리 프로그램의 최초경합을 탐지하기 위해서 수행 중에 수집된 후보사건들을 프로그램 수행 후에 분석하여 각 내포수준에서 영향을 받지 않은 경합만을 보고하는 기법을 제안한다. 분석대상이 되는 후보사건들의 접근역사에는 내포수준별로 후보사건들이 구분되며, 각 내포수준에서 독립적으로 경합을 보고한다. 이를 위해서 현재 내포수준에서 하위 내포수준의 후보사건들을 포함하여 최초경합의 가능성이 있는 사건들을 선택한다. 프로그램 수행 후에는 이렇게 선택된 후보사건들을 최상위 내포수준부터 병행성 관계를 검사하여 각 내포수준에서 경합을 탐지하고, 현재 내포수준에서 발생한 경합에 의해 영향받은 상위 내포수준의 경합을 무효화한다. 그리고 하위 내포수준에서의 최초경합 존재 여부를 판단하여 접근역사 분석과정의 종료여부를 결정한다.

제안된 기법은 각 내포수준에서 경합간의 영향관계를 고려하여 최초경합을 탐지하므로, 최초경합이 보고된 그 내포수준까지는 그 경합이 영향받지 않은 것임을 보장한다. 이는 현재 내포수준에서 최초경합 탐지 여부를 결정할 수 있게 하므로, 상위 내포수준에 대한 재분석이 필요없는 효율적인 탐지기법이다. 내포병렬성을 가진 프로그램에서 후보사건들만 수집되면, 본 기법은 수행 후에 각 공유변수에 대해 내포수준마다 최악의 경우에 $O(T^2)$[1]의 시간적 복잡성을 가지고 최초경합을 탐지한다. 따라서 본 기법은 매 접근사건마다 이러한 복잡성을 가지는 기존의 기법에 비해서 현실적이고 효율적인 디버깅을 가능하게 한다.

본 기법을 실험하기 위해서 리눅스 운영체제하의 알파 프로세서 컴퓨터에서 대상 프로그램 모델인 OpenMP[6, 7] 프로그램을 위해 Omni OpenMP 컴파일러[8]를 설치하였다. 본 기법의 구현을 위해서는 C언어를 사용하였다. 실험을 위해서 먼저, 분석해야 하는 내포깊이(N)를 정하고 각 경우에 대해 스레드 당 생성되는 스레드 수를 나타내는 단위 병렬성(t)의 크기를 증가시키면서 최초경합 보고시간을 측정하였다. 실험 결과, 최초경합의 보고시간은 최대 병렬성에 의존적이지만, 최대 병렬성 T = 160000인 경우에도 보고시간이 약 17초 정도로서 실용적임을 알 수 있었다.

2절에서는 공유메모리 병렬프로그램에서 존재하는 최초경합에 대해서 소개하고, 최초경합 탐지를 위해서 후보사건을 이용하는 기존의 연구들을 살펴보고, 이러한 방법으로 최초경합을 탐지하는 기존 기법들의 문제점을 제기한다. 3절에서는 이러한 기존 기법의 문제점에 대한 해결책으로 본 연구에서 제시하는 접근역사 분석기법을 위한 새로운 개념과 최초경합을 탐지하는 알고리즘을 정당성 증명과 함께 보인다. 그리고 4절에서는 제안된 기법의 효율성을 실험한 결과를 보이고, 마지막 절에서 결론을 제시한다.

2. 연구배경

본 절에서는 내포병렬성이 있는 프로그램에서 최초로 발생하는 경합을 소개하고, 이러한 경합의 가능성을 제공하는 후보사건들을 수집하여 최초경합을 탐지하는 기존의 기법들을 살펴본다. 그리고 이 기법들이 가진 문제점들을 분석한다.

2.1 병렬프로그램의 최초경합

본 논문은 내포병렬성을 가진 공유메모리 프로그램을 대상으로 한다. 그림 1은 OpenMP[6,7]로 작성된 병렬프로그램으로서 'PARALLEL DO'는 병렬 스레드들의 생성(fork)을 나타내고, 'END PARALLEL DO'는 병렬 스레드들의 합류(join)를 의미한다. 'SHARED(X)'로써 X를 공유변수로 지정하고, 'PRIVATE(I, J)'로써 I와 J를 각 스레드에서 사용하는 지역변수로 지정한다. 병렬프로그램의 수행은 방향성이 있는 비순환 그래프인

1) 여기서 T는 대상 프로그램의 최대 병렬성을 나타낸다.

```
C 2 3 4 5 6 7
C$OMP  PARALLEL DO SHARED(X)
C$OMP&PRIVATE(I, J)
        DO I = 1, 2
           ... = X
        IF (I .EQ. 1) THEN
           ... = F1(X) + F2(X)
        END IF
        IF (I .EQ. 2) THEN
C$OMP  PARALLEL DO
        DO J = 1, 2
           ... = N(X) + N2(X)
            X = ...
C$OMP  END PARALLEL DO
        END IF
C$OMP  END PARALLEL DO
```

그림 1 OpenMP 프로그램

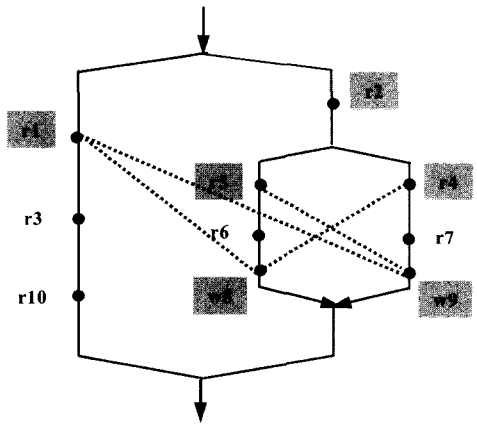

그림 2 POEG의 예

POEG(Partial Order Execution Graph)[9]으로 나타낼 수 있다. POEG에서의 정점(vertex)은 병렬 스레드의 생성과 합류를 나타내며, 정점들 사이의 간선(arc)은 정점에서 생성된 스레드를 나타낸다. 그리고 각 스레드에 접근하는 읽기와 쓰기 사건을 점으로 표시하고, r과 w로 나타내어 그 유형을 구분한다. 여기서 사용되는 숫자는 특정 수행 시에 접근사건들의 발생 순서를 의미한다. 그림 2는 그림 1에서 보인 예제 프로그램을 POEG으로 나타낸 것이다. POEG은 병렬프로그램의 수행 시에 스레드들 간의 부분적 순서(partial order)관계를 나타낸다. 두 사건사이에 경로가 존재하면 선행(happened-before) 관계가 있고, 두 사건이 순서화(ordered)되었다고 한다. 그러나 경로가 존재하지 않으면 이 두 사건은 병행(concurrent)하다. 그리고 임의의 사건 e_i가 e_j보다 선행하고 $\{e_i, e_j\}$의 각 내포수준이 $\{m, n\}$일 때, $m < n$을 만족하면 e_j가 e_i에 내포(nested)된다고 한다. 그림 2에서 $r2$와 $w9$는 그들 사이에 경로가 존재하므로 $r2$는 $w9$에 선행하고, $w9$의 내포수준이 2로서 $r2$의 내포수준 1보다 크기 때문에 $w9$는 $r2$에 내포된다. 그러나 $r4$와 $w8$ 사이에는 경로가 존재하지 않으므로 서로 병행하다.

이렇게 병행관계에 있는 두 개의 접근사건이 적어도 하나의 쓰기 사건을 포함하고 하나의 공유변수에 대해 적절한 동기화가 없이 나타날 때 발생하는 오류를 경합(race)이라고 하며, e_i-e_j로 표시한다. 여기에서 임의의 사건 e_h가 e_j에 선행하고 e_h가 경합 R_h에 포함되면, e_j는 e_h나 R_h에 의해 영향받은(affected) 사건이라고 한다. 임의의 경합을 구성하는 두 사건 $\{e_i, e_j\}$중에서 어느 사건도 다른 사건들에 의해 영향받은 사건이 아니면, 이러한 경합을 영향받지 않은(unaffected) 경합이라고 하고, 두 사건들 중에서 하나만 다른 사건에 의해 영향받으면 부분적으로 영향받은(partially affected) 경합이라 한다. 이 때, 서로 부분적으로 영향받는 경합들이 두 개 이상인 경합집합을 얽힘(tangle)이라 하고, 얽힘에 속한 임의의 경합을 얽힌(tangled) 경합이라 한다. 또한 모든 접근사건들이 기껏해야 하나의 얽힌 경합에 영향받은 경합들의 집합인 얽힘을 최초얽힘(first tangle)이라 한다. 이러한 영향받지 않은 경합이거나 그렇지 않으면 최초얽힘을 최초경합이라 한다. 예를 들어, 그림 1과 그림 2에서 경합은 {$r1$-$w8$, $r1$-$w9$, $r3$-$w8$, $r3$-$w9$, $r4$-$w8$, $r5$-$w9$, $r6$-$w9$, $r7$-$w8$, $r10$-$w8$, $r10$-$w9$, $w8$-$w9$}등이다. 이 중에서 먼저 발생한 경합은 {$r1$-$w8$, $r1$-$w9$, $r4$-$w8$, $r5$-$w9$}이고 {$r1$-$w8$, $r1$-$w9$}는 {$r4$-$w8$, $r5$-$w9$}으로부터 영향받은 경합이므로, {$r4$-$w8$, $r5$-$w9$}만이 최초경합이며 이들은 얽힘을 이루고 있다. 이러한 최초경합을 탐지하여 제거하는 것은 이 경합에 영향받은 다른 경합들이 사라질 수 있거나 새로이 발생할 수 있기 때문에 매우 중요하다.

2.2 기존의 최초경합 탐지 연구

최초경합을 수행 중에 탐지하는 기존의 기법은 모든 접근사건을 대상으로 경합을 탐지하는 기법[2,10]과 최초경합의 가능성이 있는 후보사건만을 수집하여 경합을 탐지하는 기법[3,5]이 있다.

모든 접근사건을 수행 중에 검사하여 최초경합을 보고하는 기법에는 Race Frontier[2], RecPlay[10] 등이 있다. Race Frontier 기법은 수행 중에 처음으로 경합이 탐지된 접근사건을 설정하여, 이 사건까지 프로그램을 결정적으로 재수행하면서 최초경합을 탐지하는 기법이다. 이 기법은 프로그래머로 하여금 중단점 설정이나 반복 수행을 통한 경합탐지 작업을 추가로 요구한다. RecPlay 기법은 첫 번째 수행에서 동기화 사건만을 추적파일(trace file)에 기록하고, 두 번째 수행에서 동기화

사건들 사이의 읽기와 쓰기사건 집합을 수집 및 비교하여 경합의 존재를 확인하고, 세 번째 수행에서 최초경합을 초래한 기계명령어의 위치탐지를 시도한다. 그러므로 이 기법은 최초경합을 탐지하기 위해서 최소한 세 번의 수행을 필요로 한다.

최초경합의 가능성이 있는 후보사건만을 수행 중에 수집하여 경합을 보고하는 기법에는 내포병렬성을 대상으로 하는 기법[3]과 순서적 동기화를 가진 프로그램을 대상으로 하는 기법[5]이 있다. 내포병렬성을 대상으로 하는 기법은 수행 중에 각 내포수준마다 후보사건을 수집하여 한번의 수행으로 최초경합을 탐지한다. 그리고 순서적 동기화를 가진 프로그램을 대상으로 하는 기법은 두 번의 수행을 통하여 후보사건을 수집한다. 첫 번째 수행에서는 모든 접근사건들을 감시하여 후보사건들의 부분집합을 수집한다. 그리고 첫 번째 수행과 동일한 입력으로 수행되는 두 번째 수행에서는 첫 번째 수행의 결과로 얻어진 후보사건과 최초경합에 포함되는 후보사건만을 수집하여, 이 후보사건들 간의 경합을 최초경합으로 보고한다.

2.3 기존 연구의 문제점

본 논문의 주제는 후보사건들을 이용하여 최초경합을 탐지하는 것이다. 후보사건(candidate event)[3-5]이란 최초경합에 참여할 수 있는 접근사건들을 말한다.

정의 1 임의 후보사건 e_c에 선행하는 후보사건이 존재하는 스레드의 수가 e_c를 수행하는 스레드를 포함하여 i이면, i를 e_c의 내포수준(nesting level)이라 한다.

내포병렬성을 가진 프로그램의 임의의 내포수준에서 존재하는 후보사건으로는 비내포된(non-nested) 후보사건과 내포된(nested) 후보사건으로 나눌 수가 있다. 비내포된 후보사건으로는 읽기 후보사건, 쓰기 후보사건, 비내포된 읽기-쓰기 후보사건 등이 있고, 내포된 후보사건에는 내포된 쓰기 후보사건과 내포된 읽기-쓰기 후보사건이 있다. 먼저, 임의 내포수준 i에서 어떤 접근사건 a_i에 선행하는 임의의 다른 읽기 혹은 쓰기 사건 a_j가 존재하지 않으면, a_i를 읽기 혹은 쓰기 후보사건이라 한다. 그리고 그 내포수준에서 어떤 쓰기 사건 w_i에 선행하는 다른 쓰기 사건이 존재하지 않고, w_i에 선행하는 읽기 후보사건 r_i가 존재한다면, w_i를 비내포된 읽기-쓰기 후보사건이라 한다. 마지막으로 하위 내포수준 j (> i)에서 존재하는 쓰기 후보사건과 읽기-쓰기 후보사건이 내포수준 i의 읽기 후보사건에 내포되면, 이 두 내포된 후보사건을 각각 i 내포수준의 내포된 쓰기 후보사건과 내포된 읽기-쓰기 후보사건이라 한다. 여기서 내포된 후보사건은 반드시 선행하는 i 내포수준의 읽기 후보사건이 필요하므로 비내포된 읽기-쓰기 후보사건과 함께 i 내포수준에서의 읽기-쓰기 후보사건이라 한다. 이러한 후보사건들의 최초경합에 대한 유효성을 살펴보면, 임의 내포수준 i에서의 읽기 혹은 쓰기 후보사건은 항상 유효(effective)하다. 그러나 읽기-쓰기 후보사건은 그 내포수준에 병행하는 다른 쓰기 후보사건이 존재하지 않을 경우에만 유효하다.

그림 2는 내포병렬성이 있는 병렬프로그램의 수행에서 나타난 총 10개의 접근사건들 중에서 6개의 후보사건을 음영부분으로써 보인다. {$r1$, $r2$}는 선행하는 후보사건이 존재하는 스레드가 존재하지 않으므로 (정의 1)에 의해서 내포수준 1에서의 사건이며, 같은 스레드 내에 선행하는 읽기사건이 존재하지 않으므로 내포수준 1의 읽기 후보사건이다. 사건 {$r4$, $r5$}는 자신들이 존재하는 스레드와 이들에 선행하는 읽기 후보사건 $r2$가 존재하는 스레드를 합쳐서 그 수가 2이므로, (정의 1)에 의해서 {$r4$, $r5$} 후보사건은 내포수준 2에서 존재하는 사건이다. 그리고 그 내포수준의 같은 스레드 내에 선행하는 읽기 사건이 존재하지 않으므로, 그 사건들은 내포수준 2에서의 읽기 후보사건이다. {$w8$, $w9$}는 읽기 후보사건인 {$r4$, $r5$}와 같이 내포수준 2의 사건으로서, 선행하는 읽기 사건 {$r4$, $r5$}가 존재하고 쓰기 후보사건이 존재하지 않으므로 유효한 읽기-쓰기 후보사건이다. 그리고 내포수준 1에서 보면, {$w8$, $w9$}에 선행하는 읽기 후보사건 $r2$가 존재하므로 내포된 읽기-쓰기 후보사건이다.

그림 1과 그림 2에서 나타난 후보사건들로써 구성한 접근역사는 표 1과 같다. 표 1의 세로축은 내포수준마다 후보사건들을 유형별로 수집한 것으로서, 읽기 후보사건 집합(R), 쓰기 후보사건 집합(W), 읽기-쓰기 후보사건 집합(RW) 등으로 구성된다. 그리고 읽기-쓰기 후보사건 집합(RW)은 비내포된 읽기-쓰기 후보사건 집합(RW_o), 내포된 쓰기 후보사건 집합(W_n), 내포된 읽기-쓰기 후보사건 집합(RW_n) 등으로 구성된다. 그럼에서, CS(i)는 내포수준 i에서의 후보사건 집합(candidate set)을 의미한다. 내포수준 1에는 읽기 후보사건 {$r1$, $r2$}와 읽기-쓰기 후보사건 중에서 내포된 읽기-쓰기 후보사건 {$w8$, $w9$}가 존재하고, 내포수준 2에는 읽기 후보사건 {$r4$, $r5$}와 비내포된 읽기-쓰기 후보사건 {$w8$, $w9$}가 존

표 1 후보사건들의 접근역사

후보사건유형 \ 내포수준		CS(1)	CS(2)	CS(3)
R		r1, r2	r4, r5	
W				
RW	RW_o		w8, w9	
	W_n			
	RW_n	w8, w9		

재한다. 만약, 이 수준의 읽기사건 {*r*4, *r*7}이 존재하지 않으면, 쓰기사건 *w*9가 내포수준 2의 쓰기 후보사건이 되고 내포수준 1의 내포된 쓰기 후보사건이 된다. 그리고 내포수준 3에는 후보사건이 존재하지 않는다.

정의 2 임의의 경합이 두 내포수준 {*i*, *j* | *i* ≤ *j*}에서 존재하는 두 접근사건 {e_i, e_j}을 포함하면, 이 경합을 내포수준 *i*의 경합이라 한다.

이러한 접근역사를 위해서 기존의 기법[3,5]을 사용하면, 내포수준 1의 경합 {*r*1-*w*8, *r*1-*w*9}과 내포수준 2의 경합 {*r*4-*w*8, *r*5-*w*9}을 최초경합으로 보고한다. 그러나 내포수준 1에서 존재하는 경합들은 내포수준 2에서 존재하는 경합에 영향받는 경합이므로 최초경합이 아니다. 이와 같이, 기존의 기법으로 보고된 최초경합들은 경합들간의 영향관계를 고려하지 않고 후보사건들간의 병행성 관계만을 검사하여 보고된 것이므로 최초경합임을 보장하지 못한다.

3. 접근역사 분석

본 절에서는 후보사건들을 이용하여 최초경합을 탐지하는 기존 기법들의 문제점을 해결하기 위해서 접근역사 분석기법을 제안한다. 이 기법은 유효한 후보사건들간의 경합인 후보경합(candidate race) 중에서 최초경합만을 탐지하기 위한 새로운 기법으로서, 이를 위한 접근역사 분석 알고리즘을 그 정당성 증명과 함께 보인다.

3.1 후보경합 탐지

프로그램 수행 후에 수집된 후보사건들을 이용하여 최초경합을 탐지하는 접근역사 분석기법은 후보사건을 내포수준별로 분석하여 각 내포수준에서 최초경합만을 보고한다. 따라서 각 내포수준의 후보사건과 경합에 대한 유형이 분석기준이 된다.

정의 3 임의 내포수준에서 존재하는 유효한 후보사건들간의 경합 중에서, 적어도 하나의 영향받지 않은 유효한 후보사건을 포함하는 경합과 얽힘을 각각 그 내포수준의 후보경합(candidate race)과 후보얽힘(candidate tangle)이라 한다.

정의 4 임의 내포수준 *i*에서 임의 후보경합 R_i가 존재한다고 하자. 이 때 *i* 내포수준의 임의 후보경합으로부터 영향받지 않으면서 R_i에 영향을 주는 내포수준 *j* (≥ *i*)에서의 후보경합 R_j가 존재할 수 없으면, R_i를 *i* 내포수준의 확정된(definite) 후보경합 또는 확정된 경합이라 한다.

정의 5 임의 내포수준에서 확정된 경합이 아닌 임의 후보경합이 존재하면, 그 경합을 그 내포수준의 미확정된(presumed) 후보경합 또는 미확정된 경합이라 한다.

정의 6 임의 내포수준에서 임의의 확정된 경합이나 영향받지 않은 미확정된 경합을 그 내포수준의 유효한(effective) 경합이라 한다.

예를 들어, 그림 2에서 존재하는 후보경합은 {*r*1- *w*8, *r*1-*w*9, *r*4-*w*8, *r*4-*w*9}이다. 경합{*r*1-*w*8, *r*1-*w*9}는 내포수준 1에서의 경합으로 R_1으로 표기하고, 적어도 하나의 영향받지 않은 유효한 후보사건을 포함함으로 (정의 3)에 의해서 내포수준 1에서의 후보경합이다. 경합 {*r*4-*w*8, *r*4-*w*9}는 부분적으로 서로 영향받는 경합들이므로 얽힘을 이루고, 내포수준 2에서의 경합으로 R_2로 표기한다. R_2는 (정의 3)에 의해서 내포수준 2에서의 후보얽힘이다. 후보경합 R_1과 후보얽힘 R_2에 대한 경합의 유형에 대해서 살펴보자. 후보경합 R_1은 경합을 이루는 후보사건 {*w*8, *w*9}가 내포된 후보사건으로서 이 경합들에 영향을 주는 하위 내포수준에서의 R_2가 존재하므로 (정의 4)에 의한 조건을 만족하지 못하여 확정된 경합이 아니다. 그러므로 (정의 5)에 의해서 R_1은 내포수준 1에서의 미확정된 경합이고, R_1에 속한 경합은 하위 내포수준의 R_2에 영향받으므로 내포수준 1에서의 영향받은 미확정된 경합이다. 후보얽힘 R_2는 이에 영향을 주는 후보경합이 존재하지 않으면서 내포된 후보사건이 없으므로, R_2는 하위 내포수준의 후보경합에 의해 영향받을 가능성이 없다. 그러므로 R_2에 속한 경합은 (정의 4)에 의해서 내포수준 2의 확정된 경합으로 최초얽힘을 이룬다. 그림 2의 예에서는 내포수준 2에서의 최초얽힘만이 영향받지 않은 경합으로, (정의 6)에 의해서 R_2만이 그 내포수준의 유효한 경합이다.

3.2 최초경합 탐지

본 절에서는 후보경합 중에서 최초경합만을 탐지하기 위한 접근역사 분석 알고리즘을 설명한다. 이를 위한 알고리즘 3은 쓰기 후보사건이 포함된 후보경합을 탐지하는 CheckWrite(*X*, *L*, *c*) 프로시저와 읽기 후보사건과의 후보경합을 탐지하는 CheckRead(*X*, *L*, *c*) 프로시저를 호출한다. 여기에서 매개변수로 전달되는 값들은 임의의 공유변수를 의미하는 *X*, 현재 내포수준 *L*, 현재 수행되는 사건 *c* 등이다.

알고리즘 1은 쓰기 후보사건과의 후보경합을 탐지하는 프로시저이다. 2번 줄에서는 경합의 발생 여부를 나타내는 *f_race* 변수를 거짓(false)으로 초기화한다. (3-4)번 줄에서는 매개변수로 받은 현재 쓰기 후보사건인 *c*와 후보사건 집합 중에서 읽기 후보사건과 쓰기 후보사건들인 *e*와의 병행성관계를 검사한다. 두 사건이 병행하면 5번 줄에서 현재의 쓰기 후보사건과의 경합을 확정된 경합으로 보고하며, 6번 줄에서 *f_race* 변수를 참(true)으로 갱신한다. 만일 병행하지 않으면, 9번 줄에서 거짓과 함께 반환된다.

알고리즘 2는 읽기 후보사건과의 후보경합을 탐지하는 프로시저이다. 1번 줄에서는 순서관계를 의미하는

알고리즘 1. 쓰기 후보사건과의 경합탐지

```
1: Procedure CheckWrite(X, L, c)
2:   f_race := false;
3:   for all e ∈ CS(X, L, R) ∪ CS(X, L, W) do
4:       if not ordered(c, e) then
5:           report <c, e> to DefiniteRace(X);
6:           f_race := true;
7:       endif
8:   endfor
9:   return f_race;
10: End CheckWrite
```

알고리즘 2. 읽기 후보사건과의 경합탐지

```
1: Procedure CheckRead(X, L, c)
2:   f_ordered, f_race := false;
3:   for all e ∈ CH(X, L, RW) do
4:       if not ordered(c, e) then
5:           report <c, e> to TempRace(X, RW);
6:           f_race := true;
7:       else f_ordered := true;
8:   endfor
9:   if f_ordered and f_race then
         /* tangled races */
10:      DefiniteRace(X) := TempRace(X);
11:      clear TempRace(X);
12:      return true;
13:  endif
14:  if not f_ordered and f_race then
15:      PresumedRace(X) := TempRace(X);
16:      clear TempRace(X);
17:  endif
18:  return false;
19: End CheckRead
```

f_ordered 변수와 경합의 발생여부를 나타내는 *f_race* 변수를 각각 거짓으로 초기화한다. (3-4)번 줄에서는 매개변수로 받은 현재 읽기사건인 *c*와 후보사건집합 중에서 내포된 읽기-쓰기사건(rw_o), 내포된 사건(w_n, rw_n)들인 *e*와의 병행성관계를 검사한다. (6-7)번 줄에서는 두 사건이 병행하면 *f_race* 변수를 참으로 갱신하고, 그렇지 않으면 *f_ordered* 변수를 참으로 갱신한다. 읽기 후보사건과의 경합은 확정된 경합일 수도 있고 미확정된 경합일 수도 있다. 9번 줄에서 {*f_race, f_ordered*}변수들을 검사하여 각각 모두 참이면, 이 경합을 얽힘인 확정된 경합으로 보고하고 참과 함께 반환된다. 그렇지 않으면, 14번 줄에서 *f_ordered* 변수의 값이 거짓이고 *f_race* 변수의 값이 참이면, 보고된 경합을 미확정된 경합으로 보고하고 마지막 18번 줄에서 이에 속하지 않는 경우와 같이 거짓과 함께 반환된다.

알고리즘 3. 접근역사 분석

```
1: Program HistoryAnalyzer(X)
2:   f_definite := false;
3:   for L := 1 up to Nesting_Depth do
4:       for all c ∈ CH(X, L, W) do
5:           if CheckWrite(X, L, c) then
6:               f_definite := true;
7:       endfor
8:       if f_definite then
9:           delete PresumedRace(X, RWn);
10:          exit;
11:      endif
12:      for all c ∈ CH(X, L, R) do
13:          if CheckRead(X, L, c) then
14:              f_definite := true;
                 /* tangled races */
15:      endfor
16:      if f_definite then
17:          delete PresumedRace(X);
18:          exit;
19:      endif
20:      if CH(X, L, Wn) ∪ (X, L, RWn) = ∅ then
21:          exit;
22:  endfor
23: End HistoryAnalyzer
```

알고리즘 3은 2번 줄에서 확정된 경합의 유무를 나타내는 *f_definite* 변수를 거짓으로 초기화한다. 이 변수는 이후에 쓰기 후보사건이 포함된 경합과 얽힘에 의한 확정된 경합이 탐지되면 참으로 갱신되며, 최초경합 보고를 종료하는 조건으로 사용된다. 3번 줄에서 나타나는 for문은 최대 내포깊이(*N*)만큼 수행되고, 확정된 경합이 보고되거나 더 이상 경합이 보고될 내포수준이 없으면 수행을 중단한다. for문 내부에서 수행되는 부분은 크게 세 부분으로 나눌 수 있다.

첫 번째 부분은 4번 줄에서 10번 줄까지이다. (4-6)번 줄에서는 알고리즘 1의 CheckWrite()를 쓰기 후보사건의 수만큼 수행하고, 쓰기 후보사건이 포함된 후보경합이 발생되어 참이 반환되면 확정된 경합의 발생을 의미하는 *f_definite* 변수를 참으로 갱신한다. (7-10)번 줄에서 *f_definite* 변수의 값이 참이면, 이전 내포수준까지 보고된 미확정된 경합 중에서 현재의 쓰기 사건이 포함된 경합에 영향받아 무효화되는 경합인 비내포된 읽기-쓰기 후보사건과의 미확정된 경합을 제거하므로써, 최초경합만을 보고하고 분석을 종료한다. 그러나 *f_definite* 변수의 값이 참이 아니면, 두 번째 부분을 수행한다.

두 번째 부분은 11번 줄에서 17번 줄까지이다. (11-13)번 줄에서는 읽기 후보사건의 수만큼 CheckRead()를 수행하고, 쓰기 후보사건이 존재하지 않는 내포수준

에서 얽힌 경합이 발생하면, 이러한 얽힘에 속하는 경합은 확정된 경합이므로 *f_definite* 변수를 참으로 갱신한다. (14-17)번 줄에서는 얽힘에 속하는 확정된 경합이 존재하는 경우를 위해서 그 내포수준에서의 영향받은 후보경합인 미확정된 경합을 모두 제거하고 접근역사 분석을 종료한다.

마지막으로 세 번째 부분은 18번 줄과 19번 줄이다. 이 부분은 첫 번째 부분에서의 쓰기 후보사건이 포함된 확정된 경합이 존재하지 않거나, 두 번째 부분에서의 얽힘에 의한 확정된 경합이 존재하지 않는 경우에 수행한다. 18번 줄에서 내포된 후보사건인 w_n이나 rw_n이 존재하지 않으면 더 이상 내포된 경합이 존재하지 않으므로, 19번 줄에서 최초경합 탐지를 위한 접근역사 분석을 종료한다. 임의 내포수준에서 이 부분을 수행하여 종료하면, 그 내포수준에서는 확정된 경합이 존재하지 않고 미확정된 경합만이 존재한다. 그리고 내포된 후보사건이 존재하지 않으면, 이 미확정된 경합은 영향받지 않은 경합이므로 그 내포수준의 최초경합으로 보고된다.

예를 들어, 그림 2의 후보사건들로 구성된 접근역사인 표 1에서 최초경합을 탐지해 보자. 먼저, 내포수준 1에서 쓰기 후보사건이 존재하지 않으므로 쓰기 후보사건과의 경합을 탐지하는 첫 번째 부분을 수행하지 않는다. 읽기 후보사건 {$r1$, $r2$}가 존재하므로 읽기 후보사건과의 경합을 탐지하는 두 번째 부분을 수행하고, 내포된 읽기-쓰기 후보사건 {$w8$, $w9$}와의 경합 R_1 = {$r1$-$w8$, $r1$-$w9$}를 보고한다. 이 후보경합은 얽힘이 아니므로 미확정된 경합이다. 확정된 경합이 존재하지 않고 미확정된 경합만이 존재하므로 세 번째 부분에서 내포된 후보사건의 유무를 검사한다. 내포된 후보사건 {$w8$, $w9$}가 존재하므로, 내포수준 2에서 다시 이 알고리즘을 수행한다. 내포수준 2에서 얽힘에 의한 확정된 경합 R_2 = {$r4$-$w8$, $r4$-$w9$}이 존재하므로 이에 영향받은 R_1을 무효화한다. 그리고 확정된 경합인 R_2가 존재하므로, 최초경합 탐지를 위한 접근역사 분석을 종료한다. 그러므로 이 예에서는 확정된 경합의 집합인 내포수준 2의 얽힘만이 최초경합이다.

이와 같이, 후보사건들의 병행성 관계뿐만 아니라 경합들간의 영향관계까지 고려하여 최초경합을 결정하고자 하는 본 기법은 각 내포수준에서 탐지되는 후보경합을 확정된 경합과 미확정된 경합으로 구분하고, 현재 내포수준까지에서 존재하는 확정된 경합과 영향받지 않은 미확정된 경합을 최초경합으로 결정한다. 따라서 디버깅 대상이 되는 내포병렬성을 가진 프로그램에서 후보사건들만 수집되면 본 기법은 각 내포수준에서 효율적으로 최초경합을 탐지하면서 상위 내포수준에 대한 재분석이 필요없다. 그리고 수행 후 하나의 공유변수에 대해서 최대 병렬성을 T로 할 때, 본 기법의 효율성은 각 내포수준마다 최악의 경우에도 $O(T^2)$의 시간적 복잡성으로 최초경합만을 탐지하므로, 매 접근사건마다 이러한 복잡성을 가지는 기존의 기법에 비해 현실적이고 효율적인 디버깅 기법이다.

3.3 정당성 증명

보조정리 1 임의 내포수준에서 쓰기 후보사건이 포함된 후보경합이 존재하면, 그 경합은 그 내포수준에서의 확정된 경합이다.

증명: 쓰기 후보사건이 포함된 후보경합은 쓰기 후보사건과 유효한 후보사건들 간의 경합만을 의미하므로, 쓰기 후보사건간의 경합과 쓰기 후보사건과 읽기 후보사건간의 경합 외에는 없다. 여기서, 읽기 후보사건과 쓰기 후보사건은 내포된 후보사건이 아니므로 하위 내포수준과 무관하고, 유효한 후보사건들이므로 그 내포수준에 다른 선행하는 후보사건이 존재하지 않는다. (정의 4)에 의해서 임의 내포수준에서 임의 후보경합이 존재하고, 그 내포수준의 임의 후보경합으로부터 영향받지 않으면서 그 경합에 영향을 주는 후보경합이 그 내포수준을 포함한 하위 내포수준에서 존재할 수 없으면, 그 경합은 그 내포수준의 확정된 경합이다. 그러므로 이 정리는 만족된다. □

정리 1 임의 내포수준에서 쓰기 후보사건이 포함된 확정된 경합이 존재하면, 그 내포수준의 비내포된 읽기-쓰기 후보사건이 포함된 모든 미확정된 경합은 영향받은 경합이다.

증명: 임의 내포수준 i에서 쓰기 후보사건 w_i가 포함된 확정된 경합 Rd_i가 존재하면, 내포수준 i에서는 쓰기 후보사건간의 경합과 쓰기 후보사건과 읽기 후보사건간의 경합 외에는 존재하지 않는다. 이 후보사건들이 발생 후에 나타날 수 있는 후보사건은 그 내포수준에서의 읽기-쓰기 후보사건으로 읽기 후보사건 후에만 나타난다. 그리고 내포수준 i에서의 비내포된 읽기-쓰기 후보사건 rw_i가 포함된 미확정된 경합 Rp가 존재하면, rw_i는 반드시 읽기 후보사건과 후보경합 Rp를 구성한다. 그리고 (정의 5)에 의해서 Rp가 존재하는 내포수준의 임의의 경합으로부터 영향받지 않으면서, Rp에 영향을 주는 후보경합이 그 내포수준을 포함한 하위 내포수준에서 존재할 수 있다. 그러나 Rp에 포함되는 rw_i는 같은 내포수준에서 w_i가 존재하므로, 그 내포수준의 읽기 후보사건 r_i와 후보경합을 구성할 수는 없다. 그러므로 임의 상위 내포수준 h(< i)에서 내포된 읽기-쓰기 후보사건인 rw_i와 후보경합 Rp를 이루는 읽기 후보사건 r_h가 반드시 존재한다. 그리고 rw_i는 w_i와 r_i간의 확정된 경합 Rd_i에 영향받으므로, rw_i가 포함된 미확정된 경합 Rp는 Rd_i에 영향받은 경합이다. 그러므로 이 정리는 만족된

다. □

보조정리 2 임의 내포수준에서 쓰기 후보사건이 포함된 확정된 경합이 존재하지 않고 얽힘에 속하는 경합이 존재하면, 그 경합은 그 내포수준에서의 확정된 경합이다.

증명: 임의 내포수준 i에서 쓰기 후보사건이 포함된 확정된 경합이 존재하지 않으면, 그 내포수준에는 읽기 후보사건과 읽기-쓰기 후보사건들만이 존재한다. 그리고 얽힘은 부분적으로 서로 영향받은 경합들이 두 개 이상인 임의 집합을 의미하고, 부분적으로 영향받은 경합은 임의의 경합에 영향받는 후보사건이 하나만 포함된 경합이다. 이러한 후보사건은 그 내포수준의 읽기-쓰기 후보사건으로서 이 사건에 선행하는 읽기 후보사건이 포함된 경합에 영향받는다. 그리고 이 읽기 후보사건이 포함된 경합 또한 부분적으로 영향받은 경합이므로 그 내포수준의 읽기-쓰기 후보사건이 포함된 경합일 수밖에 없다.

그리고 하위 내포수준 $j(> i)$에서 후보경합이 있다고 가정하자. 그러면, 경합의 정의에 의해서 이 경합은 그 내포수준의 읽기-쓰기 후보사건들이 포함된 경합이고, 이 경합을 구성하는 후보사건들에 공통적으로 선행하는 i 내포수준의 읽기 후보사건이 존재한다. 따라서 내포수준 j에서의 후보경합은 내포수준 i에서 존재하는 얽힘에 완전히 영향받은 경합이다. (정의 4)에 의해서 임의 내포수준에 임의 후보경합이 존재하고, 그 내포수준의 다른 임의 후보경합으로부터 영향받지 않으면서 그 경합에 영향을 주는 후보경합이 그 내포수준을 포함한 하위 내포수준에서 존재할 수 없으면, 그 경합은 그 내포수준의 확정된 경합이다. 그러므로 이 정리는 만족된다. □

정리 2 임의 내포수준에서 얽힘에 의한 확정된 경합이 존재하면, 그 내포수준의 후보사건이 포함된 모든 미확정된 경합은 영향받은 경합이다.

증명: 얽힘은 부분적으로 서로 영향받은 경합들이 두 개 이상인 임의 집합을 의미하고, 부분적으로 영향받은 경합은 임의의 경합에 영향받는 후보사건이 하나만 포함된 경합이다. 이러한 후보사건은 그 내포수준의 읽기-쓰기 후보사건으로서 이 사건에 선행하는 읽기 후보사건이 포함된 경합에 영향받는다. 임의의 내포수준 i에서 얽힘에 의한 확정된 경합 Rd_i가 존재하면, (정의 4)에 의해서 Rd_i가 존재하는 내포수준의 임의의 후보경합으로부터 영향받지 않으면서 Rd_i에 영향을 주는 후보경합이 그 내포수준을 포함한 하위 내포수준에서 존재할 수 없다. 그러므로 내포수준 i에는 쓰기 후보사건이 포함된 확정된 경합이 존재할 수 없고, 읽기 후보사건 r_i와 그 내포수준의 읽기-쓰기 후보사건 rw_i들 간의 경합만 존재한다. 여기서 rw_i들은 반드시 선행하는 읽기 후보사건이 존재하지만, 읽기 후보사건은 그 스레드에 단독(only)으로 존재할 수 있다.

그리고 내포수준 i의 후보사건이 포함된 미확정된 경합 Rp가 존재하면, Rp는 후보사건 rw_i와 병행하는 임의의 읽기 후보사건과의 경합이고 (정의 5)에 의해서 Rp가 존재하는 내포수준의 임의의 후보경합으로부터 영향받지 않으면서, Rp에 영향을 주는 후보경합이 그 내포수준을 포함한 하위 내포수준에서 존재할 수 있다. 그러나 Rp에 포함되는 rw_i는 이에 선행하는 읽기 후보사건과 함께 얽힘에 포함되므로, Rp는 rw_i와 병행하는 그 내포수준의 단독 읽기 후보사건 or_i와의 경합이다. 여기서 Rp에 포함되는 rw_i가 비내포 사건인 경우에는 임의의 상위 내포수준 $h(< i)$에서 내포된 읽기-쓰기 후보사건이므로, 이 사건과 후보경합 Rp를 이루는 단독 읽기 후보사건 or_h가 반드시 존재한다. 그러므로 얽힘에 의한 확정된 경합 Rd_i가 존재하는 내포수준 i에서의 미확정된 경합 Rp는 rw_i-or_i인 경우와 rw_i-or_h인 경우 외에는 없다. 두 경우 모두 그 내포수준의 rw_i를 포함하고, rw_i는 선행하는 읽기 후보사건과 얽힘에 의한 확정된 경합 Rd_i에 영향받으므로, rw_i가 포함된 미확정된 경합 Rp는 Rd_i에 영향받은 경합이다. 그러므로 이 정리는 만족된다. □

정리 3 임의 내포수준에서 유효한 경합의 집합이 존재하고 내포된 후보사건이 존재하지 않으면, 그 내포수준의 경합집합에 포함되지 않은 최초경합은 없다.

증명: 임의 내포수준에서 유효한 경합의 집합이 존재하면, (정의 6)에 의해서 확정된 경합의 집합이나 영향받지 않은 미확정된 경합의 집합이 존재한다. 임의 내포수준에서 확정된 경합이 존재하면, (정의 4)에 의해서 그 경합은 그 내포수준의 후보경합으로서 그 내포수준의 다른 임의의 후보경합으로부터 영향받지 않으면서 그 경합에 영향을 주는 후보경합이 그 내포수준을 포함한 하위 내포수준에서 존재할 수 없다.

그리고 임의 내포수준에서 미확정된 경합이 존재하면, (정의 5)에 의해서 그 경합은 그 내포수준의 후보경합으로서 그 내포수준의 다른 임의의 후보경합으로부터 영향받지 않으면서 그 경합에 영향을 주는 후보경합이 그 내포수준을 포함한 하위 내포수준에 존재할 수 있다. 그러나 이러한 미확정된 경합 중에서 영향받지 않은 미확정된 경합이 존재하면 이 경합에 영향을 주는 후보경합이 그 내포수준을 포함한 하위 내포수준에서 존재하지 않는다.

그러므로 확정된 경합과 영향받지 않은 미확정된 경합은 그 내포수준의 다른 임의의 후보경합으로부터 영향받지 않으므로 그 내포수준의 최초경합이다. 그리고 내포된 후보사건이 존재하지 않으면, 내포된 쓰기 후보

사건과 내포된 읽기-쓰기 후보사건이 존재하지 않으므로 하위 내포수준에 존재하는 유효한 후보경합은 없다. 그러므로 이 정리는 만족된다. □

4. 효율성 실험

실험을 위한 프로그램의 제어구조를 결정하기 위해서, 스레드 당 생성되는 스레드 수를 의미하는 단위 병렬성(t)을 모두 동일하게 두었다. 변수를 위한 접근 사건은 하나의 공유변수에 대해 생성된 모든 스레드에서 읽기 접근사건이 발생되게 하였으며, 보고수준을 조절하기 위해서 검사해야 하는 마지막 내포수준에서 읽기-쓰기 접근사건이 발생하도록 하여 종료조건인 얽힘이 발생되도록 하였다.

그림 3은 그림 2에서 보이는 후보사건 접근역사의 입력 화일 형태이다. X는 공유변수를 의미하고, 2는 내포깊이를 나타내며, {*R, W, RWO, WN, RWN*}는 후보사건들의 유형을 나타낸다. 그리고 접근사건들간의 병행성 관계를 검사하기 위한 정보인 레이블은 대상 프로그램이 내포병렬성을 가지는 경우에 효율적인 NR Labeling[4] 기법으로 생성하였다. 그림에서 [1,1,<1, 25>]는 NR Labeling으로 생성된 레이블 중에서 접근역사에 저장되는 부분이다. 따라서 그림 3은 공유변수 X에 대해 검사되는 내포깊이가 2인 추적파일로서, 내포수준 1에서 읽기 후보사건 {$r1$, $r2$}가 발생하고 내포수준 2에서 읽기 후보사건 {$r5$, $r6$}과 비내포된 읽기-쓰기 후보사건 {$w9$, $w10$}이 발생하였다.

본 기법의 효율성 실험은 쓰기 사건들의 발생 유형에 의존적이다. 그러므로 분석해야 하는 내포깊이(N)를 쓰기 사건이 발생되는 내포수준으로 하고, 내포깊이 {2, 4}에 대해 단위 병렬성(t)의 크기를 5씩 증가시키면서 최초경합의 보고시간을 측정하였다. 보고시간의 측정단위는 1/1000 초로 하였다. 표 2은 본 기법을 이용하여 실험한 결과이다. 그 결과 최초경합 보고시간이 최대 병

```
X 2
L 1
R       [1,1,<1,25>]      // r1
R       [1,1,<26,50>]     // r2
RWN     [1,1,<26,38>]     // w8
RWN     [1,1,<39,50>]     // w9

L 2
R       [1,1,<39,50>]     // r4
R       [1,1,<26,38>]     // r5
RW      [1,1,<26,38>]     // w8
RW      [1,1,<39,50>]     // w9
```

그림 3 입력 화일

표 2 접근역사 분석 시간

검사깊이 (N)	단위병렬성 (t)	최대병렬성 (T)	보고시간 (ms)
2	5	25	1
	10	100	8
	15	225	22
	20	400	54
4	5	625	118
	10	10000	699
	15	50625	5145
	20	160000	17073

렬성(T)에 의존적이지만, T = 160000인 {N = 4, t = 20} 경우에도 보고시간이 약 17초 정도로서 실용적임을 알 수 있었다.

5. 결 론

내포병렬성을 가진 공유메모리 병렬프로그램에서 수행 중에 수집된 후보사건으로 최초경합을 탐지하는 기존의 기법은 보고된 경합이 최초경합임을 보장하지 못하는 문제점이 있다. 본 연구에서는 수행 중에 수집된 후보사건들을 프로그램 수행 후에 분석하여 각 내포수준에서 확정된 경합과 영향받지 않은 미확정된 경합만을 최초경합으로 보고하는 기법을 제안하였다.

제안된 접근역사 분석기법은 임의 내포수준까지 탐지된 경합이 영향받지 않은 것임을 보장하므로, 상위 내포수준에 대한 재분석이 필요가 없는 효율적인 최초경합 탐지기법이다. 따라서 대부분의 프로그램에서 나타나는 내포병렬성에서 후보사건들만 수집되면 최초경합을 탐지할 수 있으므로, 기존의 기법에 비해서 현실적이고 효과적인 디버깅을 가능하게 한다. 향후과제로는 록킹(locking)을 가진 내포병렬성에서도 후보사건들의 접근역사를 분석하여 수행 중에 최초경합을 효율적으로 탐지하는 것이 필요하다.

참 고 문 헌

[1] Netzer, R. H. B., and B. P. Miller, "What Are Race Conditions? Some Issues and Formalizations," *Letters on Programming Language and Systems*, 1(1): 74~88, ACM, March. 1992.

[2] Choi, J., and S. L. Min, "Race Frontier: Reproducing Data Races in Parallel Program Debugging," *3rd Symp. on Principles and Practice of Parallel Programming*, pp. 145~154, ACM, April 1991.

[3] Jun, Y., and C. E. McDowell, "On-the-fly Detection of the First Races in Programs with Nested Parallelism," *2nd Int'l Conf. on Parallel and Distributed Processing Techniques and Applica-*

tions, CSREA, Sunnyvale, Calif., Aug. 1996.
[4] Kim, J., D. Kim., and Y. Jun, "Scalable Visualization for Debugging Races in OpenMP Programs," *3rd Int'l Conf. on Communications in Computing*, pp. 259~265, Las Vegas, Nevada, June 2002.
[5] Park, H., and Y. Jun, "Detecting the First Races in Parallel Programs with Ordered Synchronization," *6th Int'l Conf. on Parallel and Distributed Systems*, pp. 201~208, IEEE, Tainan, Taiwan, Dec. 1998.
[6] Chandra R., L. Dagum, D. Kohr, D. Maydan, J. McDonald, and R. Menon, *Parallel Programming in OpenMP*, Academic Press, 2001.
[7] OpenMP Architecture Review Board, *OpenMP Fortran Application Program* Interface, Ver. 2.0, Nov. 2000.
[8] Sato, M., S. Satoh, M. Kusano, and Y. Tanaka, "Design of OpenMP Compiler for an SMP Cluster," *1st European Workshop on OpenMP*, Lund, Sweden, Sept. 1999.
[9] Dinning, A., and E. Schonberg, "An Empirical Comparison of Monitoring Algorithms for Access Anomaly Detection," *2nd Symp. on Principles and Practice of Parallel Programming*, pp. 1~10, ACM, 1990.
[10] Ronsse, M., and K. De Bosschere, "RecPlay: A Fully Integrated Practical Record/Replay System," *Tr. on Computer Systems*, 17(2): 133~152, ACM, May 1999.

강 문 혜
2001년 경상대학교 컴퓨터과학과 졸업(학사). 2003년 경상대학교 대학원 컴퓨터과학과 졸업(석사). 2003년~현재 경상대학교 대학원 컴퓨터과학과 박사과정

김 영 주
1999년 경상대학교 컴퓨터과학과 졸업(학사). 2001년 경상대학교 대학원 컴퓨터과학과 졸업(석사). 2001년~현재 경상대학교 대학원 컴퓨터과학과 박사과정

전 용 기
1980년 경북대학교 컴퓨터공학과 졸업(학사). 1982년 서울대학교 컴퓨터공학부 졸업(석사). 1993년 서울대학교 컴퓨터공학부 졸업(박사). 1982~1985년 한국전자통신연구소 연구원. 1985년~현재 경상대학교 컴퓨터과학과 교수, 컴퓨터·정보통신연구소 연구원