(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2014년08월20일
(11) 등록번호 10-1432328
(24) 등록일자 2014년08월13일

(51) 국제특허분류(Int. Cl.)
G06F 15/16 (2006.01) *G06F 9/06* (2006.01)
(21) 출원번호 10-2009-7007032
(22) 출원일자(국제) 2007년11월19일
심사청구일자 2012년10월25일
(85) 번역문제출일자 2009년04월06일
(65) 공개번호 10-2009-0085028
(43) 공개일자 2009년08월06일
(86) 국제출원번호 PCT/US2007/085149
(87) 국제공개번호 WO 2008/064185
국제공개일자 2008년05월29일
(30) 우선권주장
11/602,092 2006년11월20일 미국(US)
(56) 선행기술조사문헌
US05469367 A
전체 청구항 수 : 총 13 항

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
브라운 주니어, 알렌 엘.
미국 98052-6399 워싱턴주 레드몬드 원 마이크로소프트 웨이
(74) 대리인
제일특허법인

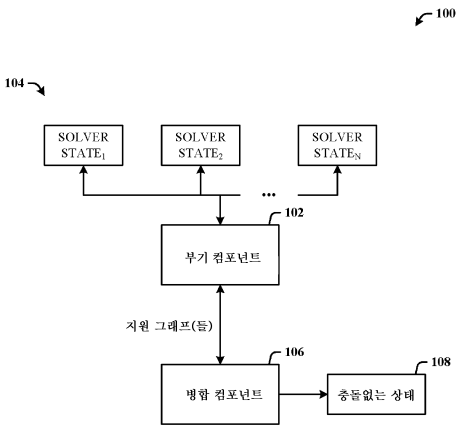
심사관 : 이석주

(54) 발명의 명칭 **제약조건 솔버 처리를 용이하게 해주는 컴퓨터 구현 시스템, 컴퓨터 구현 제약조건 솔버 처리 방법, 및 컴퓨터 구현 솔버 시스템**

(57) 요 약

병렬 CSP(constraint satisfaction problem) 솔버에서의 솔버 상태 병합(solver state merging). 병렬 CSP 솔버의 계산 쓰레드의 처리 동안의 솔버 상태는 일련의 지원 그래프(support graph)로 표현된다. 지원 그래프들이 쌍으로 병합하여 새로운 충돌없는 그래프(conflict-free graph)를 산출한다. 이 병합 프로세스는 사이클(cycle)이 없으며, 충돌이 제거되고, 쓰레드 처리가 잠금이 없다(lock-free). 이 아키텍처는 일반적으로 어떤 형식 속성(formal property)을 갖는 임의의 CSP 솔버(예를 들어, 부울 SAT 솔버)에서 적용될 수 있다. 솔버 처리를 용이하게 해주는 시스템이 제공되며, 이 시스템은 계산 쓰레드의 입력 솔버 상태를 일련의 그래프로서 표현하는 부기 컴포넌트(bookkeeping component) 및 상기 일련의 그래프 중 적어도 2개의 그래프를 쌍으로 병합하여 계산 쓰레드의 최종 상태를 표현하는 병합된 그래프를 산출하는 병합 컴포넌트(merge component)를 포함한다.

대 표 도 - 도1



특허청구의 범위

청구항 1

제약조건 솔버 처리(constraint solver processing)를 실행하는 컴퓨터 구현 시스템으로서,

계산 쓰레드의 입력 솔버 상태(input solver state)를 그래프 세트(a set of graph)로서 표현하는 부기 컴포넌트(bookkeeping component) - 상기 입력 솔버 상태는 상기 계산 쓰레드에서 동작하는 병렬 솔버로부터 수신됨 - ;

상기 그래프 세트 중 적어도 두 개의 입력 그래프를 쌍으로 병합하여 (pairwise merging) 상기 계산 쓰레드의 최종 상태를 표현하는 병합된 그래프를 산출하는 병합 컴포넌트; 및

상기 병합된 그래프에서 완전성(completeness)을 달성하기 위한 제약조건을 전파하는 전파 컴포넌트(propagation component)- 상기 전파 컴포넌트는 중간 가정(intermediate assumption)을 변경하지 않고 이전의 가정을 직접 변경하도록 제약조건 전파의 일부로서 비연대순 역추적(non-chronological backtracking)을 실행하며, 상기 전파 컴포넌트는 상기 병렬 솔버에 대한 상기 병합된 그래프의 제약조건 전파 중에 둘 이상의 학습된 제약조건을 부가하는 것을 실행하며, 상기 병렬 솔버는 병렬 충족가능성(SAT:satisfiability) 솔버임 - 를 포함하고,

상기 병합 컴포넌트에 의해 병합될 지원 그래프의 입력 세트는 충돌없는 지원 그래프(conflict-free support graphs)인

제약조건 솔버 처리를 실행하는 컴퓨터 구현 시스템.

청구항 2

제1항에 있어서,

상기 부기 컴포넌트는 상기 계산 쓰레드에서 동작하는 병렬 부울 충족가능성(SAT) 솔버로부터 상기 입력 솔버 상태를 수신하는

제약조건 솔버 처리를 실행하는 컴퓨터 구현 시스템.

청구항 3

제1항에 있어서,

상기 부기 컴포넌트는 상기 계산 쓰레드에서 동작하는 병렬 CSP(constraint satisfaction problem) 솔버로부터 상기 입력 솔버 상태를 수신하는

제약조건 솔버 처리를 실행하는 컴퓨터 구현 시스템.

청구항 4

제3항에 있어서,

상기 CSP 솔버는 평가들의 격자 A 및 값들의 집합 D에 따라 정의되며, D와 A는 동일한 집합인

제약조건 솔버 처리를 실행하는 컴퓨터 구현 시스템.

청구항 5

제1항에 있어서,

상기 병합 컴포넌트는 잠금없는 방식으로(in a lock-free manner) 상기 입력 솔버 상태를 병합하여 상기 병합된 그래프를 산출하는

제약조건 솔버 처리를 실행하는 컴퓨터 구현 시스템.

청구항 6

제1항에 있어서,

제약조건 전파 프로세스 중에 추측(guess)에 기초하여 변수 할당을 추론하는 학습 및 추론 컴포넌트를 더 포함하는

제약조건 솔버 처리를 실행하는 컴퓨터 구현 시스템.

청구항 7

제1항에 있어서,

상기 입력 솔버 상태는 상기 계산 쓰레드에서 동작하는 2개의 병렬 SAT 솔버로부터 온 것이고,

상기 병합 컴포넌트는 *L*-완전, *K*-일치 연역 그래프(*L*-complete, *K*-consistent deduction graph)의 n개(단, n은 양의 정수임)의 사본 각각에 n개의 새로운 별개의 리터럴(literal)을 추가하는 것을 실행하는

제약조건 솔버 처리를 실행하는 컴퓨터 구현 시스템.

청구항 8

컴퓨터로 구현되는 제약조건 솔버 처리 방법으로서,

계산 쓰레드의 입력 솔버 상태를 지원 그래프 세트로서 표현하는 단계,

병렬 솔버의 솔버 상태와 연관된 지원 그래프를 수신하는 단계 - 상기 솔버 상태는 상기 계산 쓰레드의 처리와 연관됨 - ,

동일한 리터럴을 갖는 노드들을 병합하기 위해 상기 병렬 솔버들 각각으로부터의 지원 그래프를 삭감하는(paring) 단계,

상기 지원 그래프를 쌍으로 병합(pairwise merging)하여 상기 계산 쓰레드의 최종 상태를 표현하는 병합된 그래프를 산출하는 단계,

중간 가정(intermediate assumption)을 변경하지 않고 이전의 가정을 직접 변경하도록 제약조건 전파의 일부로서 비연대순 역추적(non-chronological backtracking)을 사용하여 제약조건을 전파함으로써 상기 병합된 그래프의 완전성을 해결하는 단계,

상기 병렬 솔버에 대한 상기 병합된 그래프의 제약조건 전파 중에 둘 이상의 학습된 제약조건을 부가하는 단계 - 상기 병렬 솔버는 병렬 SAT 솔버임 - , 및

상기 병합된 그래프의 충돌하는 리터럴을 처리하여 충돌없는 상기 병합된 그래프를 만드는 단계를 포함하는

제약조건 솔버 처리 방법.

청구항 9

제8항에 있어서,

동일한 리터럴을 갖는 노드들에 따라 쌍으로 병합하기 위해 상기 지원 그래프를 삭감하는 단계를 더 포함하는

제약조건 솔버 처리 방법.

청구항 10

제8항에 있어서,

개개의 상기 병렬 솔버의 상기 지원 그래프를 동시에 처리하는(manipulating) 단계를 더 포함하는

제약조건 솔버 처리 방법.

청구항 11

제8항에 있어서,

충돌하는 리터럴 세트로부터 하나의 충돌하는 리터럴을 제거하고 상기 충돌하는 리터럴 세트의 나머지 충돌하는 리터럴을 새로운 연역 그래프에 대한 가정(assumption)으로서 지정하는 단계를 더 포함하는

제약조건 솔버 처리 방법.

청구항 12

제8항에 있어서,

상기 병렬 솔버들은 부울 SAT 솔버인

제약조건 솔버 처리 방법.

청구항 13

컴퓨터 구현 솔버 시스템으로서,

계산 쓰레드의 입력 솔버 상태를 지원 그래프 세트로서 표현하는 컴퓨터 구현 수단,

병렬 CSP 솔버들로부터 상기 지원 그래프를 수신하는 컴퓨터 구현 수단 - 상기 지원 그래프는 상기 계산 쓰레드의 처리와 연관된 병렬 솔버 상태를 표현함 -,

상기 병렬 CSP 솔버들 각각으로부터의 지원 그래프를 삭감하는 컴퓨터 구현 수단,

상기 지원 그래프를 쌍으로 병합하여 병합된 그래프를 산출하는 컴퓨터 구현 수단,

상기 병합된 그래프에서 충돌을 제거하여 상기 계산 쓰레드의 최종 상태를 표현하는 충돌없는 병합된 그래프를 출력하는 컴퓨터 구현 수단,

중간 가정(intermediate assumption)을 변경하지 않고 이전의 가정을 직접 변경하도록 제약조건 전파의 일부로서 비연대순 역추적(non-chronological backtracking)을 사용하여 제약조건을 전파함으로써 상기 병합된 그래프의 완전성을 해결하는 컴퓨터 구현 수단, 및

상기 병렬 솔버에 대한 상기 병합된 그래프의 제약조건 전파 중에 둘 이상의 학습된 제약조건을 부가하는 컴퓨터 구현 수단 - 상기 병렬 솔버는 병렬 SAT 솔버임 - 를 포함하는

컴퓨터 구현 솔버 시스템.

청구항 14

삭제

청구항 15

삭제

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

명 세 서

배 경 기 술

[0001] 프로세서, 메모리 및 저장 장치 등의 하드웨어에서의 기술적 진보가 계속하여 많은 서로 다른 종류의 미디어 데이터 유형(예를 들어, 음성, 텍스트 및 비디오), 개발 프로그램, 기타 등등을 처리함으로써 더 풍부한 사용자 경험을 제공하는 더 크고 더 복잡한 소프트웨어 애플리케이션을 생성하는 촉매로서 역할하고 있다.

[0002] 무어의 법칙과 연관된 과거의 회로 속도 향상이 더 이상 쉽게 달성가능하지 않은 것으로 보이기 때문에, 단일-프로세서 시스템에서 이들 벤더가 의존하는 하드웨어 지원이 요원할 수 있다. 무어의 법칙의 주된 측면은, 일반적으로 장치 제조에서의 기술적 진보로 인해, 대략 18개월마다 칩 상의 트랜지스터의 수가 2배로 된다는 것이다. 과거에는, 이것이 달성되었을 때, 프로세서 클럭 속도도 역시 증가될 수 있었다. 그렇지만, 현재 더 조밀하게 집적된 트랜지스터와 연관된 열밀도가 높아서 클럭 속도를 증가시키는 것은 열이 효율적이고 효과적으로 방출될 수 없다는 것을 의미한다. 따라서, 소형 장치가 더 이상 곧바로 더 빠르고 열을 덜 발생하면서 동작하는 기계가 되지는 않는다.

[0003] 이용되는 한가지 다른 대안은 단순히 장치들을 더 많이 사용하는 것이다. 환언하면, 예를 들어, 프로세서 분야에서는, 소프트웨어 요구를 수용하기 위해 병렬 또는 멀티-프로세서 시스템을 설계한다. 그렇지만, 병렬 처리 시스템은 알고리즘 또는 계산 쓰레드 처리를 취급하는 고도의 조정 기법을 필요로 한다. 제약조건 풀기(constraint solving)는 이들 조정 기법들을 테스트하는 데 유용하다. 그렇지만, 종래의 순차적 알고리즘은 모든 이용가능한 공유 메모리 병렬 프로세서를 효과적으로 사용하도록 재구성하기가 어렵기로 악명이 높다.

[0004] 부울 SAT(Boolean satisfiability) 솔버 등의 CSP(constraint satisfaction problem) 솔버도 결코 이상의 관찰 내용에 대한 예외가 아니다. 통상적으로, 순차 CSP 솔버는 부분해(partial solution)(제약조건 변수들 중 일부에 대한 할당)를 포함하는 현재 상태를 가지며, 이 현재 상태로부터 솔버는 하나 이상의 현재 미할당된 변수들을 할당함으로써 생성되는 보강해(augmented solution)를 갖는 새로운 상태로 이동하려고 시도한다. 새로운 할당은 제약조건의 전파를 통해 다른 할당들을 야기할 수 있다. 제약조건이 전파하면 차례로 (충돌을 완화시키기 위해) 부분적으로 취소(undo)되어 새로운 할당으로 변경되고 재전파되어야만 하는 현재의 할당들 간에 충돌이 검출될 수 있다.

[0005] 병렬 처리 시스템에서, 이러한 문제 해결 방식의 병렬 구현은 필연적으로 방금 기술한 방식으로 제약조건을 전파하는 몇가지 병렬 계산을 갖는다. 문제는 (전파 후) 몇개의 충돌 없는 솔버 상태를 병합하여 하나의 충돌없는 솔버 상태를 산출하는 것이다.

발명의 상세한 설명

[0006] 이하는 개시된 발명의 몇몇 측면들에 대한 기본적인 이해를 제공하기 위해 간략화된 요약을 제공한다. 이 요약은 포괄적인 개요가 아니며, 본 발명의 주요/필수 구성요소를 확인하거나 그 범위를 정하기 위한 것이 아니다. 그의 유일한 목적은 나중에 제시되는 보다 상세한 설명에 대한 서문으로서 몇몇 개념들을 간략화된 형태로 제시하는 데 있다.

[0007] 개시된 아키텍처는 일반 CSP(constraint satisfaction problem) 솔버에서 병렬 처리에 대한 지원을 제공한다. 솔버의 계산 쓰레드의 상태는 일련의 지원 그래프로서 표현된다. 일련의 지원 그래프는, 일반 CSP 솔버의 중요한 컴포넌트가 되는 경우가 많은 TMS(truth maintenance system)의 효율적인 구현에서 공인된 메카니즘이다. 본 명세서에 기술된 바와 같이, 지원 그래프는 그래프를 쌍으로 병합하여 새로운 충돌없는 그래프를 산출하는 새로운 방식으로 사용된다. 이것은 다수의 새로운 할당에 걸쳐 제약조건의 병렬 전파를 매핑하고 또 다수의 전파로부터 얻어지는 상태들을 병합하여 (더 많은 변수들이 할당되어 있는) 새로운 문제 솔버 상태로 축소시킴으로써 CSP 솔버를 구성하는 것을 가능하게 해준다. 이 아키텍처는 일반적으로 어떤 형식 속성을 갖는 임의의 CSP 솔버에서 적용될 수 있다. 예를 들어, 한 구현예에서, 이 아키텍처는 구체적으로 부울 SAT(Boolean satisfiability) 솔버와 관련하여 적용될 수 있다.

[0008] 본 명세서에 개시되고 청구된 아키텍처는 솔버 처리를 용이하게 해주는 컴퓨터 구현 시스템을 포함한다. 이 시스템은 계산 쓰레드의 입력 솔버 상태를 일련의 그래프로서 표현하는 부기 컴포넌트(bookkeeping component)를 포함한다. 병합 컴포넌트(merge component)는 일련의 그래프 중 적어도 2개의 입력 그래프를 쌍으로 병합하여 계산 쓰레드의 최종 상태를 표현하는 병합된 그래프를 산출하는 것을 수행한다.

[0009] 이상의 목적 및 관련 목적을 달성하기 위해, 개시된 발명의 어떤 예시적인 측면들이 본 명세서에서 이하의 설명 및 첨부 도면과 관련하여 기술된다. 그렇지만, 이들 측면은 본 명세서에 개시된 원리들이 이용될 수 있는 다양

한 방식들 중 단지 몇개만을 나타낸 것이며, 이러한 측면 및 그의 등가물 전부를 포함하는 것으로 보아야 한다. 다른 이점들 및 새로운 특징들이 도면과 관련하여 살펴볼 때 이하의 상세한 설명으로부터 명백하게 될 것이다.

실 시 예

[0026] 개시된 아키텍처는 종래에 순차적 문제조차도 해결하기 어렵기로 악명높았던 공유 메모리 병렬 프로세서 시스템에 대한 해결책을 제공한다. 본 발명은 병렬 CSP(constraint satisfaction problem) 솔버에서의 잠금없는(lock-free) 상태 병합을 제공한다. 솔버의 계산 쓰레드의 상태는 일련의 지원 그래프로서 표현된다. (혼동이 일어나지 않는 한, 간단함을 위해, "일련의 지원 그래프"보다는 어구 "지원 그래프"가 사용될 것이다) 이들 지원 그래프는 그래프들을 쌍으로 병합하는 새로운 방식으로 사용되며, 이 방식은 잠금이 없으면서 새로운 충돌없는 지원 그래프를 산출하는 프로세스를 말한다. 이 아키텍처는 일반적으로 기본 문제가 부울 충족가능성(Boolean satisfiability)으로 환원가능한(reducible) 임의의 CSP 솔버에서 적용될 수 있고, 한 구체적인 구현예에서, 이 아키텍처는 구체적으로 부울 SAT(Boolean satisfiability) 솔버와 관련하여 적용될 수 있다.

[0027] SAT 솔버의 구체적인 구현예에서, SAT 솔버 문제는 2가지 방식 중 하나로 문제를 야기하는 일련의 부울식(Boolean formula)으로 시작된다. 식이 항상 참인지 여부를 아는 것이 요망된다. 달리 말하면, 식이 정리(theorem)인지를 판정하는 것이 요망된다. 임의의 식의 보수(complement)(또는 부정(negation))을 취하는 것에 의한 동치 문제는 식이 충족가능(satisfiable)인지를 판정하는 것이다. 즉, 식을 참으로 만드는 변수의 할당이 있는가? 기계화를 위해, 이것은 문제에 접근하는 방식일 수 있다. 이는 문제를 증명하는 정리가 아니라 충족가능성 문제로서 착수된다.

[0028] 디지털 설계의 공간에서, 부울식이 정리인지 여부를 아는 것이 요망된다. 보여줄 식이 정리인지를 증명하기 보다는, 부정이 충족가능하지 않은지를 보여준다. 부울식이, 예를 들어, DNF(disjunctive normal form) 등의 표준형(canonical form)으로 제시된다. DNF는 식들의 집합이 유한이고 식들 중 임의의 것에 단지 2개의 논리적 연결어(logical connective)(즉, 부정 기호(논리 NOT) 및 논리합(즉, 논리 OR))만이 있는 표현이다. 따라서, 현재 표준형으로 되어 있는 원래의 문제가 충족가능하기 위해, 이들 논리합 식이 참이도록 하는 변수들의 할당이 있다. 표준형으로 된 각각의 식을 절(clause)이하고 한다. 본 명세서에서 사용되는 바와 같이, 용어 "절"은 또한 제약조건이라고도 할 수 있다.

[0029] 종종 CSP 솔버 등의 이러한 복잡한 영역에서, 부분해조차도 도출될 수 있는 분석적 방법이 없다. 대답(본질적으로, 추측)을 가정해야만 한다. 그 결과, 프로그램에 의해 일단 참인 것으로 생각된 초기 가정이 시간에 따라 변할 수 있고 나중에 거짓인 것으로 발견될 수 있다. 그에 따라, 프로그램은 나중에 거짓인 것으로 밝혀진 가정에 기초하여 하였을 수 있는 추론을 취소하는 문제점을 갖는다. 효율적이기 위해, 프로세스는 취소를 가능한 한 적게 시도한다.

[0030] SAT는 방금 기술한 문제에 직면하고 있는데, 그 이유는 충족가능성을 입증하기 위해, 대답들의 부분 할당(partial assignment)이 추측되기 때문이다. 추측은 새로운 추측의 논리적 결과로 인해 다른 부울 변수 대답(Boolean variable answer)이 할당되게 한다. 이러한 추측의 결과는 모든 변수들이 할당되게 하는 성공적인 일련의 추측들이 궁극적으로 행해지거나 행해진 마지막 추측이 논리적 모순(logical inconsistency)에 이르러, 그 지점에서 추측들 중 하나가 포기된다.

[0031] 종래에, 후방으로 이동하면서 한번에 하나씩 변수에 대한 값을 선택하기 위해 연대순 역추적(chronological backtracking)이 이용될 수 있다. 이러한 역추적은 변수에 할당할 적법한 값이 남아 있지 않을 때까지 계속될 수 있다.

[0032] 개시된 발명은 가장 관련있는 가정을 취소하는, 예를 들어, 그 하나의 가정만을 변경하고 중간 가정들에 대해서는 아무 것도 변경하지 않는 기능을 제공함으로써 "비연대순" 역추적(non-chronological backtracking)을 이용한다. 환언하면, 이 알고리즘은 임의의 경로를 따라 거꾸로 가서 하나의 가정을 선택하고 그 가정만을 변경할 수 있으며, 이와 달리 종래의 시스템에서는 역추적이 현재의 위치와 어떤 이전의 위치 사이의 모든 노드를 변경한다.

[0033] 이제부터, 본 발명에 대해 도면들을 참조하여 기술하며, 도면들 전체에 걸쳐 유사한 참조 번호가 유사한 구성요소를 지칭하는 데 사용된다. 이하의 설명에서, 설명의 목적상, 본 발명에 대한 완전한 이해를 제공하기 위해 많은 구체적인 상세가 기술되어 있다. 그렇지만, 본 발명이 이들 구체적인 상세 없이도 실시될 수 있다는 것이 분명하다. 다른 경우에, 본 발명의 설명을 용이하게 해주기 위해 공지의 구조 및 장치가 블록도 형태로 도시되

어 있다.

[0034] 먼저, 도면들을 참조하면, 도 1은 병렬 구현에 따라 솔버 처리를 용이하게 해주는 시스템(100)을 나타낸 것이다. 시스템(100)은 계산 쓰레드 병렬 솔버의 처리와 연관된 솔버 상태(104)(SOLVER $STATE_1$, SOLVER $STATE_2$,...,SOLVER $STATE_N$로 표기되어 있고, 여기서 N은 양의 정수임)를 일련의 지원 그래프로서 표현하는 부기 컴포넌트(102)를 포함한다. 솔버 상태(104)는 부기 컴포넌트(102)에 의해 처리를 위해 서로 다른 시스템으로부터 병렬 방식으로 수신될 수 있다. 부기 컴포넌트(102)는 솔버 상태(104)를 처리하여 차후의 병합을 위한 지원 그래프를 산출한다. 시스템(100)은 또한 일련의 지원 그래프들 중 적어도 2개의 입력 지원 그래프를 쌍으로 병합하여 계산 쓰레드의 최종 상태를 표현하는 병합된 그래프를 산출하는 병합 컴포넌트(106)를 포함할 수 있다.

[0035] 부기 컴포넌트(102) 및 병합 컴포넌트(106)에 의한 지원 그래프의 처리가 동시적일 수 있다. 환언하면, 솔버들 중 하나로부터의 솔버 상태가 수신되어 다른 솔버로부터의 솔버 상태의 처리 동안에 처리된다.

[0036] 한 구현예에서, 부기 컴포넌트(102)는 계산 쓰레드를 처리하고 있는 병렬 CSP 솔버들로부터 입력 솔버 상태(104)를 수신한다. 이하에서 더 상세히 기술하는 바와 같이, CSP 솔버는 평가들의 격자(lattice of valuations) A 및 값들의 집합 D(단, D 및 A는 동일한 집합임)에 따라 정의된다. 보다 구체적인 대안의 구현예에서, 부기 컴포넌트(102)는 계산 쓰레드를 처리하는 병렬 부울 SAT 솔버들로부터 입력 솔버 상태(104)를 수신한다.

[0037] 입력 솔버 상태(104)가 계산 쓰레드를 처리하는 2개의 병렬 SAT 솔버로부터 온 것인 경우, 병합 컴포넌트(102)는 n개의 새로운 서로 다른 리터럴(literal)을 L-완전, K-일치 연역 그래프(L-complete, K-consistent deduction graph)의 n개(단, n은 양의 정수임)의 사본 각각에 추가하는 것을 용이하게 해준다. 이것은 또한 이하에서 더 기술될 것이다.

[0038] 병합 컴포넌트(106)는 입력 솔버 상태를 잠금없는(lock-free) 방식으로 병합하여 사이클을 갖지 않는 병합된 그래프를 산출하고 병합된 그래프에서 충돌을 제거함으로써 충돌없는 그래프(108)를 출력한다.

[0039] 잠금없는 처리와 관련하여, 병렬성(parallelism)을 달성하는 한가지 방식은 병렬 쓰레드 처리를 담당하는 기능을 갖는 것이다. 환원하는, 궁극적으로 결과 마무리(results finalization)를 필요로 하는 다수의 병렬 쓰레드가 있다.

[0040] 병렬 쓰레드 처리를 수행하는 한가지 종래의 방식은 제1 쓰레드가 결과에 도달할 때까지 어떤 공유 데이터 구조로부터의 나머지 쓰레드들을 단순히 잠금(lock)하는 것이다. 그러면, 이전의 쓰레드 결과들과 일치하는 결과를 얻는 일이 나머지 쓰레드의 몫이 된다.

[0041] 본 발명은 병렬로 한 스텝씩 처리하여 나아가고 이어서 그 결과들을 결합함으로써 종래의 잠금을 회피한다. 그에 따라, m개(단, m은 양의 정수임)의 독립적인 에이전트가 있기 때문에, 각각의 에이전트가 한 스텝씩 나아가고, 전체적인 대답을 얻기 위해 에이전트가 결과들을 결합해야만 한다는 점에서 이 방법은 잠금이 없다. 프로세스의 핵심은 결과들이 한번에 2개씩 결합된다는 것이다.

[0042] 지원 그래프들을 병합할 때, 몇가지 바람직하지 않은 일들이 일어날 수 있다. 첫째, 2개의 그래프 사이에 충돌이 발생한다. 환언하면, 제1 그래프 내의 변수 x가 참 값을 할당받고, 다른 그래프 내의 동일 변수 x가 거짓을 할당받는다. 해결하기 위해, 2개의 그래프가 병합되고 이러한 충돌을 가져오는 가정들의 집합이 구해진다. 일단 구해지면, 연관된 가정 또는 가정들 중 하나가 철회된다.

[0043] 한 그래프에서의 할당이 가정이고 다른 그래프에서의 동일한 할당이 단위 분해(unit resolution)에 의해 도출될 때 두번째 문제가 일어날 수 있다. 이제 그래프 병합이 시도되면, 결과 그래프는 사이클을 나타낸다. 해결하기 위해, 가정이 탐욕적 방법(greedy method)을 사용하여 아주 경제적인 방식으로 포기된다. 이것에 대해서는 이하에서 더 상세히 설명된다.

[0044] 도 2는 본 발명에 따른 솔버 상태 처리 방법을 나타낸 것이다. 설명의 간단함을 위해, 예를 들어, 플로우차트 또는 흐름도의 형태로 본 명세서에 도시된 하나 이상의 방법들이 일련의 동작들로서 도시되고 기술되어 있지만, 본 발명에 따르면 몇몇 동작들이 본 명세서에 도시되고 기술된 것과 다른 순서로 및/또는 다른 동작들과 동시에 행해질 수 있기 때문에, 본 발명이 동작들의 순서에 의해 제한되지 않는다는 것을 잘 알 것이다. 예를 들어, 당업자라면 방법이 다른 대안으로서, 상태도에서와 같이, 일련의 상호관련된 상태 또는 이벤트로서 표현될 수 있다는 것을 잘 알 것이다. 게다가, 본 발명에 따른 방법을 구현하는 데 예시된 동작들 전부가 필요한 것은 아

닐 수도 있다.

[0045] 200에서, 병렬 솔버들의 솔버 상태로부터 지원 그래프가 생성된다. 202에서, 솔버 상태의 지원 그래프가 수신되고, 이 솔버 상태는 계산 쓰레드의 처리와 연관되어 있다. 204에서, 각각의 솔버로부터의 지원 그래프가 동일한 리터럴을 갖는 노드들을 병합하기 위해 동시에 삭감된다. 206에서, 지원 그래프가 쌍으로 병합되어 계산 쓰레드의 최종 상태를 표현하는 병합된 지원 그래프를 산출한다. 208에서, 병합된 그래프의 완전성(completeness)을 달성하기 위해 제약조건 전파(constraint propagation)가 개시된다. 210에서, 이전의 가정들을 변경하여 충돌을 해결하기 위해 제약조건 전파 동안에 비연대순 역추적이 이용된다. 212에서, 충돌없는 병합된 그래프를 출력하기 위해 충돌하는 리터럴이 제거된다.

[0046] 이제 도 3을 참조하면, 대답들의 할당에 대한 추측을 추리하기 위해 학습 및 추론을 이용하는 대안의 병렬 솔버 시스템(300)이 도시되어 있다. 시스템(300)은 솔버 상태를 발생하는 다수의 시스템(예를 들어, 2개의 순차 솔버 시스템)(302)을 포함한다. 예를 들어, 제1 시스템(304)(SYSTEM$_A$로 표기됨), 예를 들어, 제1 순차 CSP 솔버는 솔버 상태(306)(SOLVER STATE$_{A1}$,...,SOLVER STATE$_{AS}$로 표기되어 있으며, 여기서 S는 양의 정수임)를 발생하고, 제2 시스템(308)(SYSTEM$_B$로 표기됨), 예를 들어, 제2 순차 CSP 솔버는 솔버 상태(310)(SOLVER STATE$_{B1}$,...,SOLVER STATE$_{BT}$로 표기되어 있으며, 여기서 T는 양의 정수임)를 발생한다.

[0047] 부기 컴포넌트(102)는 솔버 상태(306, 310)를 대응하는 제1 및 제2 시스템(304, 308)로부터 수신하여 시스템 제약조건을 해명하는 병렬 쓰레드 처리를 위한 솔버 상태의 그래프(예를 들어, 지원 그래프)를 생성한다. 이 그래프는 충돌없는 상태를 갖는 병합된 그래프로 병합하기 위해 병합 컴포넌트(106)로 전달된다. 그렇지만, 병합된 그래프가 완전하지 않을 수 있다. 그에 따라, 병합된 그래프에서의 완전성을 보장하기 위해 전파 컴포넌트(312)에 의해 용이하게 되는 제약조건 전파가 이용된다. 이것에 대해서는 이하에서 더 상세히 기술한다. 전파 컴포넌트(312)는 또한 중간의 가정을 변경하지 않고 이전의 가정을 직접 변경하기 위해 제약조건 전파의 일부로서 비연대순 역추적을 용이하게 해준다.

[0048] 시스템(300)은 또한 제약조건 전파 동안에 추측에 기초한 변수 할당에 관한 추론을 용이하게 해주는 학습 및 추론 컴포넌트(314)(본 명세서에서 추론 엔진(inference engine)이라고도 함)를 이용할 수 있다.

[0049] 시스템(300)은 대답들 전부를 알고 있는 것은 아닌 애플리케이션에서 동작할 수 있다. 그에 따라, 도대체 무슨 일이 있어나고 있는지에 관한 가정이 행해진다. 예를 들어, 서버 프로그램이 더 많은 데이터에 노출될 때, 프로그램이 행한 이전의 가정들이 거짓인 것으로 밝혀지는 경우가 있을 수 있다. 그러면, 프로그램은 거짓 전제에 기초하여 행해졌을 수 있는 추론들을 취소하는 문제를 갖는다.

[0050] 충족가능성 문제에서, 유사한 문제가 존재할 수 있는데, 그 이유는 충족가능성을 증명하기 위해, 대답들의 할당이 추측되어야만 할 수도 있기 때문이다. 환언하면, 순차적 경우에서 새로운 추측이 행해질 때, 이것은 새로운 추측의 논리적 결과로 인해 다른 부울 변수 대답들이 할당되게 한다. 게다가, 2가지 일 중 하나가 궁극적으로 일어난다. 즉, 성공적인 일련의 추측이 행해지고 이로 인해 모든 변수들이 할당되게 되거나, 행해진 마지막 추측이 논리적 모순에 이르게 되고 그 지점에서 추측들 중 하나가 포기된다.

[0051] 대부분의 전통적인 형태의 부울 충족가능성에서, 마지막 추측의 부호가 변경되고, 따라서 이전에 부울 변수가 거짓을 할당받은 경우, 이 할당이 참으로 변경되고 프로세스가 다시 시작된다. 할당이 참도 아니고 거짓도 아닐 수 있을 때 논리적 모순이 일어날 수 있으며, 이는 행해진 어떤 이전의 할당이 잘못되었음에 틀림없다는 것을 의미한다.

[0052] 반복적으로 백업하고 앞으로 나아가기보다는, 필요에 따라, 충돌이 관찰되는 실제의 일련의 가정들이 식별된다. 충돌이 관찰된 경우, 무엇이 충돌을 가져오는지(행해진 마지막 가정일 수도 있고 그렇지 않을 수도 있음)를 정확하게 진단하는 것이 요망된다. 이러한 진단은 정확하게 행해질 수 있고 그 결과 일련의 모순되는 가정이 식별될 수 있다.

[0053] 본 아키텍처는 (예를 들어, 선택과 관련하여) 그의 다양한 측면들을 수행하기 위해 다양한 학습 및 추론 기반 방식을 이용할 수 있다. 예를 들어, 어느 할당을 (예를 들어, 참에서 거짓으로 또는 거짓에서 참으로) 토글할지를 결정하는 프로세스는 자동 분류기 시스템 및 프로세스를 통해 용이하게 행해질 수 있다.

[0054] 분류기는 입력 속성 벡터 x = (x1, x2, x3, x4, *xn*)를 클래스 라벨 *class(x)*에 매핑하는 함수이다. 분류기는 또한 입력이 클래스에 속할 신뢰도(confidence)를 출력할 수 있다, 즉 f(x) = *confidence*(*class*(x))이다. 이러한 분류는 사용자가 자동으로 수행되기를 원하는 동작을 예측 또는 추론하기 위해 확률적 및/또는 다른 통계

적 분석(예를 들어, 한명 이상의 사람들에 대한 기대값을 최대화하기 위해 분석 유틸리티 및 비용을 고려함)을 이용할 수 있다.

[0055] 본 명세서에서 사용되는 바와 같이, "추론한다" 및 "추론"은 일반적으로 이벤트 및/또는 데이터를 통해 포착된 일련의 관찰 결과로부터 시스템, 환경 및/또는 사용자의 상태를 추리 또는 추론하는 프로세스를 말한다. 추론은 특정의 상황 또는 동작을 식별하기 위해 이용될 수 있거나, 예를 들어, 상태들에 걸친 확률 분포를 생성할 수 있다. 추론은 확률적일 수 있다, 즉 데이터 및 이벤트의 고려에 기초하여 관심의 상태들에 걸친 확률 분포의 계산일 수 있다. 추론은 또한 일련의 이벤트 및/또는 데이터로부터 상위 레벨 이벤트를 작성하는 데 이용되는 기법을 말할 수 있다. 이벤트들이 시간상으로 아주 근접하여 상관되어 있든, 이벤트 및 데이터가 하나 이상의 이벤트 및 데이터 소스로부터 온 것이든 간에, 이러한 추론의 결과, 일련의 관찰된 이벤트 및/또는 저장된 이벤트 데이터로부터 새로운 이벤트 또는 동작이 구성된다.

[0056] 지원 벡터 기계(support vector machine, SVM)는 이용될 수 있는 분류기의 일례이다. SVM은 가능한 입력들의 공간에서 트리거링 입력 이벤트를 비트리거링 이벤트로부터 최적의 방식으로 분할하는 초곡면(hypersurface)을 찾아내는 동작을 한다. 직관적으로, 이것은 분류가 훈련 데이터와 똑같지 않은 그 근처에 있는 테스트 데이터에 대해 정확하도록 해준다. 기타 유향 및 무향 모델 분류 방식으로는, 예를 들어, 여러가지 형태의 통계적 회귀, NB(naive Bayes), 베이지안 네트워크, 결정 트리, 신경망, 퍼지 논리 모델이 있으며, 서로 다른 독립성 패턴을 표현하는 기타 통계적 분류 모델이 이용될 수 있다. 분류는 본 명세서에서 사용되는 바와 같이 등급 및/또는 우선순위를 할당하는 데 사용되는 방법들을 포함한다.

[0057] 본 명세서로부터 잘 알 것인 바와 같이, 본 아키텍처는 (예를 들어, 일반 훈련 데이터를 통해) 명시적으로 훈련되는 분류기는 물론 (예를 들어, 사용자 거동의 관찰, 외부 정보의 수신을 통해) 암시적으로 훈련되는 분류기도 이용할 수 있다. 예를 들어, SVM은 분류기 구성자(classifier constructor) 및 특징 선택 모듈 내에서의 학습 또는 훈련 단계를 통해 구성된다. 따라서, 분류기(들)는 미리 정해진 기준에 따라 다수의 함수를 자동으로 학습 및 수행하는 데 이용될 수 있다.

[0058] 환언하면, 학습 및 추론 컴포넌트(314)는 충돌 처리를 위해 학습된 제약조건을 적용할 수 있다. 일례로서, 이하에 기술되는 바와 같이, 하나의 학습된 제약조건이 충돌 해결을 위해 순차 SAT 솔버에 추가될 수 있다. 이와 유사하게, 병렬 SAT 솔버에서, 몇개의 학습된 제약조건이 제약조건 전파 처리 동안에 추가될 수 있다.

[0059] 제약조건 전파 예를 살펴보기 전에, 더 나은 이해를 위해 예비 정보가 제공된다. 제약조건 충족 문제(constraint satisfaction problem, CSP)는 통상적으로 다음과 같이 기술된다. 변수들의 유한 집합 V, (통상적으로 유한인) 값들의 집합 D, (아마도 무한인) 평가들의 격자 A 및 함수들(제약조건들)의 유한 집합 $D^{|V|} \rightarrow A$가 주어진 경우, $D^{|V|}$ 내의 튜플로서 모든 제약조건에서 그 튜플의 이미지들 간의 격자 합류(lattice meet)가 A에서 최대가 되는 튜플을 찾아낸다. 이하는 D 및 A 둘다가 부울 격자인 경우의 설명으로서, SAT 솔버와 관련하여 본 발명을 구체적으로 기술하고 있다. 이러한 제한에도 불구하고, 본 발명은 D 및 A가 동일한 집합인 임의의 CSP로 일반화가능하다.

[0060] 변수는 $x_1, x_2, \ldots, x_n$(단, n은 양의 정수임)로 표기되어 있다. 리터럴은 "부호있는" 변수(signed variable) $x_1, \neg x_1, x_2, \neg x_2, \ldots$ 이다. L은 리터럴들의 집합으로서 I은 L에 걸쳐 있다. 절(clause)은 리터럴들의 유한 집합으로서, □로 표기되는 비어있는 절(empty clause)을 포함한다. 절 $\{x_1, \neg x_2, x_3, \neg x_4\}$은 종종 그 대신에 $x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4$ (그의 원소들의 논리 OR$(\vee)$)로서 쓰여진다. 명제 논리식들의 유한 집합이 논리적으로 동치인 절들의 유한 집합으로 표현될 수 있다.

[0061] 변수가 미할당되어 있거나 True(예를 들어, x_1) 또는 False(예를 들어, $\neg x_1$)의 값을 할당받을 수 있다. 변수 x에 값 True을 할당하는 것은 리터럴 x와 융합(conflate)된다. 변수 x에 값 False를 할당하는 것은 리터럴 $\neg x_1$와 융합된다. 할당을 변수로부터 절로 격상시킨 결과로서 평가가 절에 기인한다. True의 할당을 갖는 절은 충족되고, False의 할당을 갖는 절은 위반된다. 충족가능성은 일련의 절을 충족시키는 논리적 변수들의 할당이 있는지의 문제이다. 항진 명제(tautology)는 일련의 절이 모든 할당에 의해 충족되는지의 문제이다. 충족가능성 및 항진 명제는 식의 부정이 충족가능하지 않는 한 그 식이 항진 명제라는 점에서 이중적 개념(dual concept)이다.

[0062] K가 L 내의 리터럴들에 걸친 절들의 집합이라고 하자. (L,K)-연역 그래프는 유향 그래프로서, 그의 노드(node)는 L의 부분집합들과 쌍을 이루는 개개의 리터럴로 라벨 표시되고 그의 간선(edge)은 K의 구성원으로 라벨 표시되어 있다. 어느 L 및 K를 의미하는지에 관하여 혼동이 없는 경우, 그래프는 단순히 연역 그래프(deduction graph)라고 한다. 노드는 첫번째 것에서 두번째 것으로의 유향 간선이 있고 그 간선이 절 k로 라벨 표시되어 있는 경우에만 다른 것에 대한 k-전제(k-antecedent)이다.

[0063] (L,K) 연역 그래프는 노드 상의 라벨이 그의 전제 노드들 상의 라벨들의 합집합인 경우에만 제대로 라벨 표시된다(well-labeled), 즉 노드 l이 들어오는 원호를 갖지 않는 경우, 그 노드는 $\{l\}$로 라벨 표시되고, 리터럴 l에 대한 노드로 들어오는 모든 간선은 $l \vee k$ 형태의 절로 라벨 표시되며, 리터럴 l에 대한 노드로부터 나가는 모든 간선은 $\neg l \vee k$ 형태의 절로 라벨 표시된다. l 노드에 들어오는 $l \vee l_1 \vee \ldots \vee l_m$ 로 라벨 표시된 간선이 있을 때는 언제나, 역시 l 노드에 들어오는 $l \vee l_1 \vee \ldots \vee l_m$ 로 라벨 표시된 m-1개의 다른 간선이 있다.

[0064] 자신으로 이루어진 단원 집합(singleton set)(예를 들어, $\neg x_9@\{\neg x_9\}$)으로 라벨 표시된 (L,K)-연역 그래프에서의 노드 l은 가정 리터럴(assumption literal)이라고 한다. 제대로 라벨 표시된 (L,K)-연역 그래프는 노드에 들어오는 원호가 정확히 하나의 절로 라벨 표시되는 경우에 일의적으로 정렬된다(uniquely justified). 제대로 라벨 표시된 (L,K)-연역 그래프는 노드 l 및 $\neg l$ 둘다를 포함하는 것은 아닌 경우에만 K-일치(K-consistent)이다. 제대로 라벨 표시된 (L,K)-연역 그래프는, $l \vee l_1 \vee \ldots \vee l_m$ 형태의 K 내의 모든 k에 대해 $\neg l_1,\ldots,\neg l_m$이 그래프 내에 있을 때는 언제나 l이 k로 라벨 표시된 들어오는 입력 간선(input incident edge)을 갖는 그래프 내에 있는 경우에만, K-완전(K-complete)이다. 사이클이 없는 제대로 라벨 표시된 (L,K)-연역 그래프 G는, (노드의 관점에서) 더 큰 제대로 라벨 표시된 그래프 G'가 없는 경우에, L-완전(L-complete)이다.

[0065] 도 4 및 도 5는 충돌 처리를 위해 지원 그래프를 이용하는 병렬 솔버의 예를 나타낸 것이다. 도 4는 예시적인 제약조건 전파 프로세스로부터 도출된 (L,K)-연역 그래프(400)를 나타낸 것이다. 상기 정의의 적용을 설명하기 위해, 충족가능성 문제 및 시도된 부분해의 일례를 생각해보자. 이하의 것들을 포함하는 제약조건들의 초기 집합 C

$$\omega_1 \equiv (\neg x_1 \vee x_2)$$
$$\omega_2 \equiv (\neg x_1 \vee x_3 \vee x_9)$$
$$\omega_3 \equiv (\neg x_2 \vee \neg x_3 \vee x_4)$$
$$\omega_4 \equiv (\neg x_4 \vee x_5 \vee x_{10})$$
$$\omega_5 \equiv (\neg x_4 \vee x_6 \vee x_{11})$$

[0066]

$$\omega_6 \equiv (\neg x_5 \vee \neg x_6)$$
$$\omega_7 \equiv (x_1 \vee x_7 \vee \neg x_{12})$$
$$\omega_8 \equiv (x_1 \vee x_8)$$
$$\omega_9 \equiv (\neg x_7 \vee \neg x_8 \vee \neg x_{13})$$

[0067]

[0068] 및 현재의 가정 스택(stack of assumption)으로 시작한다. 연역 그래프(400)에서, 가정들이 가정 노드(assumption node)로서 모델링되고, 이 가정 노드는 입력 원호(input arc)를 갖지 않는다. 따라서, 그래프(400)에 대한 가정들은 이하의 것들을 포함한다.

$$x_1@\{x_1\},$$
$$\neg x_9@\{\neg x_9\},$$
$$\neg x_{10}@\{\neg x_{10}\},$$

[0069] $$\neg x_{11}@\{\neg x_{11}\}.$$

[0070] 그래프(400)는 4개의 가정 노드, 즉 x_1에 참(예를 들어, x_1)이 할당되어 있는 제1 가정 노드(402), x_9에 거짓(예를 들어, $\neg x_9$)이 할당되어 있는 제2 가정 노드(404), x_{10}에 거짓(예를 들어, $\neg x_{10}$)이 할당되어 있는 제3 가정 노드(406), 및 x_{11}에 거짓(예를 들어, $\neg x_{11}$)이 할당되어 있는 제4 가정 노드(408)를 포함한다.

[0071] 시간축에서의 스냅샷을 살펴보면, 그래프(400)는 가정 노드(402)가 x_1에 대한 참(예를 들어, x_1)의 할당을 갖고, 가정 노드(404)는 x_9에 대한 거짓(예를 들어, $\neg x_9$)의 할당을 갖는다. 이제부터, 상기한 바와 같이, ω_2로 라벨 표시된 간선, 제약조건 절 $\omega_2 \equiv (\neg x_1 \vee x_3 \vee x_9)$(즉, "not x_1 or x_3 or x_9"로 읽혀질 수 있음)을 생각해보자. 그렇지만, 유의할 점은 제약조건 ω_2에서, x_1이 $\neg x_1$로서 부정으로 나타나며(이는 그래프(400)에서 참으로서 할당하는 것이 논리합 인자(disjunct)를 거짓으로 만든다는 것을 의미함), 부정으로 할당된 x_9도 역시 논리합 인자를 거짓으로 만든다는 것이다. 그렇지만, 전체적인 제약조건 ω_2는 참으로 되어야만 하며 제약조건 절 ω_2을 참으로 되게 할 수 있는 유일한 남은 방법은 x_3가 참(예를 들어, x_3)으로 할당될 때이다. 이것은 바로 그래프(400)가 노드(410)에서 x_1을 참으로 할당함으로써 나타내는 것이다. 따라서, 입력 노드(402, 404)에서의 가정 및 관련 제약조건 ω_2에 기초하여, 제약조건 ω_2를 충족시키는 유일한 방법은 x_3를 참으로 할당하는 것이다. 이것은 단위 분해(unit resolution)의 적용이다.

[0072] 계속하여 다른 노드들에서, 노드(412)로의 간선은 제약조건 $\omega_1 \equiv (\neg x_1 \vee x_2)$을 사용한다. 그렇지만, 노드(402)에서의 x_1은 참으로 할당되어 있고, 제약조건 ω_1을 충족시키는 유일한 나머지 방법은 노드(412)에서의 x_2를 참으로 할당하는 것이다. ω_3로 라벨 표시된 간선들을 살펴보면, 제약조건 절 $\omega_3 \equiv (\neg x_2 \vee \neg x_3 \vee x_4)$는 노드(414)에서의 x_4에 참을 할당함으로써만 충족될 수 있다. 환언하면, 414에의 입력 노드는 410 및 412이고, 이들 입력 노드(410 및 412)는 각각 x_3 및 x_2에 참을 할당하여, ω_3를 충족시키는 남아 있는 유일한 방법이 x_4에 참을 할당하는 것이다.

[0073] ω_4로 라벨 표시된 간선들을 살펴보면, 제약조건 절 $\omega_4 \equiv (\neg x_4 \vee x_5 \vee x_{10})$은 노드(416)에서의 x_5에 참을 할당하는 것에 의해서만 충족될 수 있는데, 그 이유는 입력 x_{10}에 거짓이 할당되고 입력 x_4에 참이 할당되기 때문이다. ω_5로 라벨 표시된 간선들은 제약조건 절 $\omega_5 \equiv (\neg x_4 \vee x_6 \vee x_{11})$를 사용하며, 이 제약조건 절은 노드(418)에서의 x_6에 참을 할당하는 것에 의해서만 충족될 수 있는데, 그 이유는 입력 x_{11}에 거짓이 할당되고 입력 x_4에 참이 할당되기 때문이다. 마지막으로, ω_6으로 라벨 표시된 간선들은 제약조건 절 $\omega_6 \equiv (\neg x_5 \vee \neg x_6)$을 사용하며, 이 제약조건은 대응하는 노드(416, 418)에서의 x_5 및 x_6에 대한 기존의 참의 할당에 기초하여 실패한다.

[0074] 제약조건 전파는 (L,K)-연역 그래프(400)를 가져오며, 마지막 노드(420)에 도달할 때, 새로 도출된 제약조건(다른 형태로 되어 있음)을 출력한다.

[0075] $$\omega_C(\kappa(\omega_6)) \equiv \neg(x_1 \wedge \neg x_9 \wedge \neg x_{10} \wedge \neg x_{11})$$

[0076] 여기서, κ(카파)는 "충돌"을 나타내고, 이 제약조건은 무언가 잘못되었다는 것의 보고이다. 여기에서, 출력 $\omega_C(\kappa(\omega_6))$는 동시에 x_1이 참이고 x_9가 거짓이며 x_{10}이 거짓이고 x_{11}이 거짓일 수는 없고 이들 중 하나가 추가의 처리를 위해 포기되어야만 한다는 것을 나타낸다. 이것은 not x_1 or x_9 or x_{10} or x_{11}과 동치이며, 이것이 올바른 형태이다. 도 5는 도 4의 지원 그래프로부터의 충돌하는 제약조건 출력을 처리하는 새로운 그래프(500)를 나타낸 것이다. 상기한 것들 4개 중 3개(예를 들어, $\neg x_9$, $\neg x_{10}$, 및 $\neg x_{11}$)에 대한 가정을 선택하면 새로운 그래프(500) 및 또다른 새로 도출된 제약조건을 산출한다.

[0077] $\omega_C(\kappa(\omega_9)) \equiv \neg(\neg x_9 \wedge \neg x_{10} \wedge \neg x_{11} \wedge x_{12} \wedge x_{13})$

[0078] 보다 구체적으로, 그래프(500)는 가정 노드(502, 504, 및 506)(각각 $\neg x_9@\{\neg x_9\}$, $\neg x_{10}@\{\neg x_{10}\}$, 및 $\neg x_{11}@\{\neg x_{11}\}$)를 사용하여 가정을 행함(이전의 가정들 중 하나를 누락시킴)으로써 생성된다. 이들 가정으로 시작하여 간선들에 새로 도출된 제약조건 $\omega_C(\kappa(\omega_6))$을 적용하면 제약조건 $\omega_C(\kappa(\omega_6))$를 충족시키기 위해 노드(508)에서의 x_1에 거짓을 할당하는 것을 산출한다.

[0079] ω_7로 라벨 표시된 간선들을 살펴보면, 다른 가정 노드(510)($x_{12}@\{x_{12}\}$를 할당받음)가 도입되고, 제약조건 절 $\omega_7 \equiv (x_1 \vee x_7 \vee \neg x_{12})$은 노드(512)에서의 x_7에 참을 할당하는 것에 의해서만 만족될 수 있는데, 그 이유는 입력 x_1에 거짓이 할당되어 있고 입력 x_{12}에 참이 할당되어 있기 때문이다.

[0080] ω_8로 라벨 표시된 간선을 살펴보면, 제약조건 절 $\omega_8 \equiv (x_1 \vee x_8)$은 노드(514)에서의 x_8에 참을 할당하는 것에 의해서만 만족될 수 있는데, 그 이유는 입력 x_1에 거짓이 할당되어 있기 때문이다. 이제, ω_9로 라벨 표시된 간선들을 생각해보면, 제약조건 절 $\omega_9 \equiv (\neg x_7 \vee \neg x_8 \vee \neg x_{13})$이 현재의 입력(노드(512), 노드(514) 및 가정 노드(516))을 사용하여 충족될 수 없다. 따라서, 노드(518)에서 출력되는 새로 도출된 제약조건 $\omega_C(\kappa(\omega_9)) \equiv \neg(\neg x_9 \wedge \neg x_{10} \wedge \neg x_{11} \wedge x_{12} \wedge x_{13})$은 연역 그래프(500)에 무언가 잘못되었다는 보고이다. 제약조건 도출 프로세스는 충돌 상태가 존재하지 않을 때까지 계속된다.

[0081] 더 일반적인 방식으로 기술하면, 각각의 병렬 솔버 쓰레드의 솔버 상태에 대해 지원 그래프가 획득되면, 병합이 시작될 수 있다. 본 설명이 쌍으로 병합하는 것에 중점을 두고 있지만, 병합이 3개 이상의 그래프를 사용하여 달성될 수 있는 것이 생각된다. 상태 병합은 잠금이 없다(lock-free). 환언하면, 병렬성을 달성하는 종래의 한 방식이 쓰레드 처리를 수반한다. 예를 들어, 다수의 병렬 쓰레드가 데이터를 처리하고 있는 경우, 궁극적으로 쓰레드의 결과를 병합하는 것이 요망된다. 이것을 하는 한가지 방법은 제1 쓰레드가 그의 프로세스를 완료할 때까지 제1 쓰레드에서 어떤 공유 데이터 구조로부터의 다른 쓰레드들을 단순히 잠금하는 것이다. 그에 따라, 이미 작성된 결과들과 일치하는 결과들을 작성하는 것은 다른 잠금된 쓰레드들의 몫이다. 이러한 종래의 아키텍처는 적어도 비효율적이다. 개시된 아키텍처는 이러한 명시적인 잠금 메카니즘을 회피한다.

[0082] 상기한 바와 같이, 알고리즘(들)은 병렬로 한 스텝씩 나아가며 처리를 수행하고 이어서 결과들이 결합된다. 환언하면, 전체적인 결과에 도달하기 위해, 각각의 에이전트는 한 스텝씩 나아가며 처리를 하고 개별적인 중간 결과들이 결합되며, 이것은 전체적인 대답에 도달할 때까지 계속된다. 그에 부가하여, 충돌 처리가 해결되고 병합 프로세스는 사이클이 없다(cycle free).

[0083] 이전과 같이, 충돌 방지 처리(deconflict processing)에서, 그래프들 중 하나 또는 다른 하나에서 하나 이상의 가정이 포기될 수 있다. 게다가, 어느 가정을 포기할지에 관하여 아주 경제적인 것이 바람직하다. 개시된 아키텍처에 의해 사용되는 정보 집합들(예를 들어, 가정, 연역, 그래프,...) 전부를 정의함으로써, 환원 알고리즘(본 명세서에서 redux라고도 함)은 병합되는 2개의 그래프에서 가능한 한 적게 포기하는 방식으로 선택을 한다. 단일 그래프의 경우에, 이들 충돌을 식별하는 것의 요점은, 몇개의 이전의 가정들에 걸쳐 백업해야 하는 것(이는 일련의 지원 그래프에 대한 요점임)보다는, 포기되는 경우 가능한 적은 영향을 갖는 가정을 식별하는 것임을 상기한다.

[0084] 이제, 한 쌍의 그래프에 동일한 기법(들)을 적용하면, 가정이 포기되는 경우 가능한 한 적은 영향 또는 손실이 있도록 하는 사이클없는 방식으로 그래프들이 병합된다. 이것은 "탐욕적" 방법 또는 알고리즘을 이용하여 해결될 수 있다. 환언하면, 이 방법은 특정의 시점에서 최선인 것처럼 보이는 선택을 함으로써 포기되는 것을 최소화하는 저렴한 방법을 찾는다. 훨씬 더 많은 분석을 이용하는 보다 집중적인 알고리즘이 이용될 수 있지만, 한 구현예에서는, 탐욕적 방법으로 충분하다.

[0085] 도 6은 2개의 지원 그래프를 작성 및 병합하는 방법을 나타낸 것이다. 600에서, 2개의 지원 그래프가 처리를 위해 수신된다. 602에서, 탐욕적 방법을 사용하여 동일한 리터럴에 대응하는 노드들을 병합해 사이클링을 축소 또는 제거하기 위해 이들 그래프가 삭감(즉, 축소)된다. 604에서, 지원 그래프를 병합하여 병합된 그래프를 산출한다. 606에서, 필요한 경우 제약조건들을 전파함으로써 완전성을 얻기 위해 병합된 그래프를 처리한다.

608에서, 필요한 경우, 충돌을 찾기 위해 병합된 그래프를 처리한다. 610에서, 충돌없는 솔버 상태를 표현하는 최종적인 병합된 그래프가 출력된다.

[0086] 2개의 (L,K)-연역 그래프가 어떻게 병합될 수 있는지에 대한 보다 상세한 설명에 위한 준비로서, 이하의 정의들이 제공된다. G, G'이 비순환 불완전(acyclic, incomplete) (L,K)-연역 그래프인 경우,

[0087] $C_{G,G'}$는 그래프 G 및 G'에 공통인 가정들의 집합이다.

[0088] A_G는 G의 가정들의 집합이다.

[0089] D_G는 G의 연역들의 집합이다.

[0090] $A_{G,G'}$은 G'의 연역인 G의 가정들의 집합이다.

[0091] $B_{G,G'}$은 G'에서 언급되지 않은 G의 가정들의 집합이다.

[0092] $$A_G = C_{G,G'} \cup A_{G,G'} \cup B_{G,G'}$$
$$A_{G'} = C_{G,G'} \cup A_{G',G} \cup B_{G',G}$$

[0093] $f_G: A_G \rightarrow 2^{L-A_G} - f_G$는 G에서의 가정들의 *종속부(dependent)*를 생성하는 함수이다.

[0094] $h_G: L\text{-}A_G \rightarrow 2^{A_G} - h_G$는 G에서의 연역의 *전제(antecedent)*를 생성하는 함수이다.

[0095] 도 7은 쌍으로 병합하는 프로세스(pairwise merging process)에서 2개의 입력 지원 그래프를 삭감하는 방법을 나타낸 것이다. 700에서, 이 특정의 구현예에서, 이 프로세스는 redux라고 하는 함수에 따라 개시된다. 함수 redux(G, G')는 다음과 같이 정의되는 한쌍의 그래프(G_1, G_2)를 생성한다. 702에서, $l \in A_{G,G}$를 $|f_G(l)|$(종속부들의 집합의 크기)가 적어도 $A_{G,G'}$에서의 임의의 다른 선택된 가정 노드보다 큰 노드라고 하는 제1 정의가 제공된다. 704에서, $l' \in A_{G',G}$를 $|f_G(l')|$가 적어도 $A_{G',G}$에서의 임의의 다른 선택된 가정 노드보다 큰 노드라고 하는 제2 정의가 제공된다. 706에서, $|f_G(l)|$가 $|f_G(l')|$보다 작은 경우, G와 G'의 역할이 반대로 된다. 708에서, $|f_{G'}(l'')|$이 l''의 임의의 다른 선택된 것보다 작은 $l'' \in h_{G'}(l)$이 선택된다. 710에서, G''을 l'' 및 l_2가 $|f_{G'}(l'')|$에 있는 임의의 노드 l_2가 없는 G'의 서브그래프인 것으로 정의한다. 712에서, $A_{G,G'} = A_{G'',G} = \varnothing$인 경우, redux를 종료하고, 쌍 (G,G'')을 반환한다. 714에서, $A_{G,G'} = A_{G'',G} = \varnothing$이 아닌 경우, redux (G,G'')를 호출한다.

[0096] 궁극적으로, redux는 동일한 리터럴에 대응하는 노드들이 사이클을 염려하지 않고 병합될 수 있도록 2개의 그래프를 삭감한다. 이하에서 설명하는 바와 같이, redux는 가능한 한 적은 도출된 (비가정 리터럴)을 포기하려고 한다는 점에서 "탐욕적" 방식으로 삭감한다. 종국에 가능한 한 많은 병렬 전방향 진행(parallel forward progress)이 달성되는 방식으로 삭감하는 것이 바람직하다. 그렇지만, 사이클을 염려하지 않고 병합하기 위해 그래프를 삭감하는 다른 알고리즘들이 이용될 수 있다는 것을 잘 알 것이다.

[0097] 유의할 점은 병합된 그래프(또다시 G라고 함)가 더 이상 L-완전(L-complete)이 아니며, 이는 교정을 하도록 제약조건을 전파함으로써 교정될 수 있다. 또한, 병합된 그래프가 충돌하는 리터럴 l 및 $\neg l$을 포함할 수도 있으며, 이 경우에 이 그래프에 대해 deconflict(G)가 계산되어 충돌없는 그래프를 생성한다. 도 8은 병합된 그래프의 충돌을 처리하는 방법을 나타낸 것이다. 800에서, 이 특정의 구현예에서, deconflict라고 하는 함수에 따라 충돌을 제거하는 일이 개시된다. 802에서, $|f_{G'}(l')|$이 임의의 다른 이러한 선택보다 작은 $l' \in h_G(l)$을 선택한다. 804에서, $|f_{G'}(l'')|$이 임의의 다른 이러한 선택보다 작은 $l'' \in h_G(\neg l)$를 선택한다. 806에서, $|f_{G'}(l')| \leq |f_{G'}(l'')|$인 경우, 삭제될 가정으로서 그의 종속부 전부와 함께 l을 선택한다(그렇지 않은 경우,

l'을 선택한다). 808에서, 모든 충돌하는 노드가 제거될 때까지 이 프로세스가 반복된다.

[0098] redux 및 deconflict 함수는 redux를 한쌍의 그래프에 먼저 적용하고 이어서 그 결과에 deconflict를 적용하는 병합 함수를 가정함으로써 SAT 솔버에서 이용된다. 도 9는 지원 그래프를 삭감 및 병합하여 순차 SAT 솔버에 대한 최종적인 병합된 충돌없는 그래프를 산출하는 방법을 나타낸 것이다. 900에서, 순차 SAT 솔버에서 그래프 삭감(예를 들어, redux를 사용함) 및 충돌 처리(예를 들어, deconflict를 사용함)가 개시된다. 902에서, 순차 SAT 솔버의 내부 루프는 하나의 새로운 리터럴을 L-완전, K-일치 연역 그래프(L-complete, K-consistent deduction graph)에 추가한다. 904에서, 노드를 병합하여 사이클없는 처리를 생성하기 위해 지원 그래프가 삭감된다. 906에서, 병합된 그래프를 L-완전으로 만들기 위해 제약조건이 전파된다. 908에서, 선택적으로 하나의 학습된 제약조건을 추가함으로써 (예를 들어, 함수 deconflict를 사용하여) 충돌없는 병합된 그래프가 제공된다. 910에서, 필요에 따라 솔버 상태의 병합된 충돌없는 그래프를 출력하기 위해 이 프로세스가 반복된다.

[0099] 도 10은 지원 그래프를 삭감 및 병합하여 병렬 SAT 솔버에 대한 최종적인 병합된 충돌없는 그래프를 산출하는 방법을 나타낸 것이다. 1000에서, 병렬 SAT 솔버에서 그래프 삭감(예를 들어, redux를 사용함) 및 충돌 처리(예를 들어, deconflict를 사용함)가 개시된다. 1002에서, 병렬 SAT 솔버의 내부 루프는 L-완전, K-일치 지원 그래프의 n개의 사본 각각에 n개의 새로운 서로 다른 리터럴을 추가한다. 1004에서, 노드를 병합하여 사이클없는 처리를 생성하기 위해 n개의 그래프가 삭감된다. 1006에서, 병합된 그래프를 L-완전으로 만들기 위해 각각의 연역 그래프 내에서 제약조건이 전파된다. 1008에서, 선택적으로 학습된 제약조건을 추가함으로써 (예를 들어, 함수 deconflict를 사용하여) 충돌없는 병합된 그래프가 출력된다.

[0100] 도 11은 본 발명의 솔버 상태 처리를 멀티-코어 처리 시스템(1102)에 적용하는 시스템(1100)을 나타낸 도면이다. 멀티-코어 처리 시스템(1102)은 제1 처리 코어(1104)($CORE_1$로 표기됨) 및 제2 처리 코어(1106)($CORE_2$로 표기됨)(둘다 동일한 다이 상에 제조됨)에 의해 용이하게 될 수 있는 공유 메모리 병렬 처리 시스템이다. 처리 시스템(1002)은 또한 처리되고 있는 계산 쓰레드의 공유 버퍼링을 위한 온보드 메모리(1108)를 포함할 수 있다. 동일한 다이 상에 도시되어 있지만, 메모리(1108)가 다이 외부에 위치할 수 있으며 동일한 목적을 다할 수 있다.

[0101] 각각의 코어(1104, 1106)에 의한 공유 쓰레드 처리를 지원하기 위해, 한쌍의 솔버가 제공된다. 예를 들어, 프로세서 시스템(1102)의 다수의 코어(1104, 1106) 간의 쓰레드 실행 동안에 제약조건 처리를 수행하기 위해 제1 솔버(1110)($SOLVER_1$로 표기됨) 및 제2 솔버($SOLVER_2$로 표기됨)(예를 들어, 둘다 CSP 솔버일 수 있음)가 제공된다.

[0102] 본 발명에 따라 지원 그래프를 쌍으로 처리하기 위해 상태 시스템(1114)이 제공된다. 상태 시스템(1114)은 지원 그래프의 형태로 연관된 병렬 솔버(1110, 1112)로부터 수신되는 솔버 상태($SOLVER_1$ STATE 및 $SOLVER_2$ STATE로 표기됨)를 쌍으로 처리하는 것을 제공한다. 상태 시스템(1114)는 지원 그래프 축소(즉, 삭감), 지원 그래프 병합, 충돌 처리 및 변수 할당을 달성하기 위해 상기한 부기 컴포넌트(102), 병합 컴포넌트(106), 전파 컴포넌트(312) 및 추론 컴포넌트(314)를 포함할 수 있다.

[0103] 상태 시스템(1114)이 순전히 소프트웨어로, 순전히 하드웨어로[예를 들어, ASIC(application specific integrated circuit) 장치 또는 FPGA(field programmable gate array)로서], 또는 하드웨어와 소프트웨어 둘다의 조합으로서 구현될 수 있다는 것을 잘 알 것이다. 다른 대안으로서, 상태 시스템(1114)의 컴포넌트(102, 106, 312, 314)는 하드웨어 및/또는 소프트웨어의 조합으로서 개별적으로 구현될 수 있다.

[0104] 도 12는 본 발명에 따른 솔버 상태 처리를 멀티-코어 멀티-프로세서 시스템에 적용하는 시스템(1200)을 나타낸 도면이다. 멀티-코어 멀티-프로세서 시스템(1200)은 제1 멀티-코어 프로세서 시스템(1202) 및 제2 멀티-코어 프로세서 시스템(1204)(이들 각각은 공유 메모리 병렬 처리 시스템임)을 포함한다. 제1 프로세서 시스템(1202)은 제1 처리 코어(1206)($CORE_1$로 표기됨) 및 제2 처리 코어(1208)($CORE_2$로 표기됨)를 포함하며, 이들 둘다는 동일한 다이 상에 제조되고 처리되고 있는 계산 쓰레드의 공유 버퍼링을 위해 공유 메모리(1208)를 공유한다.

[0105] 제2 멀티-코어 프로세서 시스템(1204)은 제1 처리 코어(1212)($CORE_1$로 표기됨) 및 제2 처리 코어(1214)($CORE_2$로 표기됨)를 포함하며, 이들 둘다는 동일한 다이 상에 제조되고 처리되고 있는 계산 쓰레드의 공유 버퍼링을 위해 공유 메모리(1216)를 공유한다.

[0106] 각각의 코어(1206, 1208)에 의한 공유 쓰레드 처리를 지원하기 위해, 대응하는 솔버 집합이 제공된다. 예를 들

어, 프로세서 시스템(1202)의 다수의 코어(1206, 1208) 간의 쓰레드 실행 동안에 제약조건 처리를 수행하기 위해 제1 솔버(1218)($SOLVER_1$로 표기됨) 및 제2 솔버(1220)($SOLVER_2$로 표기됨)(예를 들어, 둘다 CSP 솔버일 수 있음)가 제공된다. 이와 유사하게, 각각의 코어(1212, 1214)에 의한 공유 쓰레드 처리를 지원하기 위해, 대응하는 솔버 집합이 제공된다. 예를 들어, 프로세서 시스템(1204)의 다수의 코어(1212, 1214) 간의 쓰레드 실행 동안에 제약조건 처리를 수행하기 위해 제3 솔버(1222)($SOLVER_3$로 표기됨) 및 제4 솔버(1224)($SOLVER_4$로 표기됨)(예를 들어, 둘다 CSP 솔버일 수 있음)가 제공된다.

[0107] 각각의 솔버(1218, 1220, 1222 및 1224)는 솔버 상태를 상태 처리 시스템(1226)에 전달한다. 예를 들어, 솔버 상태(S_1 STATE 및 S_2 STATE로 표기됨)는 제1 프로세서 시스템(1202)에서 상태 시스템(1226)으로 전달되고, 솔버 상태(S_3 STATE 및 S_4 STATE로 표기됨)는 제2 프로세서 시스템(1204)에서 상태 시스템(1226)으로 전달된다.

[0108] 본 발명에 따라 지원 그래프를 쌍으로 처리하기 위해 상태 시스템(1226)이 제공된다. 상태 시스템(1226)은 병렬 솔버(1218, 1220, 1222 및 1224)로부터 지원 그래프의 형태로 수신되는 솔버 상태를 쌍으로 처리하는 것을 제공한다. 예를 들어, 이러한 멀티-코어 멀티-프로세서 시스템에서, 쓰레드 계산 처리가 코어들(1206, 1208, 1212 및 1214)의 임의의 조합에 걸쳐 일어날 수 있다. 예를 들어, 코어(1208, 1212 및 1214)를 사용하여 처리가 행해질 수 있다. 그에 따라, 이들 3개의 코어로부터의 상태가 개시된 알고리즘에 따라 쌍으로 처리하기 위해 상태 시스템(1226)으로 전달되어야만 한다. 이를 지원하기 위해, 상태 시스템(1226)은 상태 처리를 받고 있는 쓰레드에 기초하여 각각의 솔버(1218, 1220, 1222 및 1224)로부터 관련된 상태를 선택하는 선택 컴포넌트(1228)를 포함할 수 있다. 환언하면, 관련없는 상태는 지원 그래프 처리를 위해 선택되지 않는다. 그렇지만, 하나의 상태 시스템이 각각의 프로세서 시스템(1202 또는 1204)에 전용되도록 부가적인 상태 시스템(도시 생략)을 포함함으로써 상태 처리가 이제 병렬로도 수행될 수 있다는 것을 잘 알 것이다.

[0109] 상기한 바와 같이, 상태 시스템(1226)은 지원 그래프 축소(즉, 삭감), 지원 그래프 병합, 충돌 처리 및 변수 할당을 달성하기 위해 상기한 부기 컴포넌트(102), 병합 컴포넌트(106), 전파 컴포넌트(312) 및 추론 컴포넌트(314)를 더 포함할 수 있다.

[0110] 그에 부가하여, 상태 시스템(1226)이 순전히 소프트웨어로, 순전히 하드웨어로(예를 들어, ASIC, FPGA), 또는 하드웨어와 소프트웨어 둘다의 조합으로서 구현될 수 있다는 것을 잘 알 것이다.

[0111] 상태 시스템(도 11의 1114 및/또는 1226)은 또한 솔버 상태의 처리를 위해 컴퓨팅 시스템에 설치하는 독립형 소프트웨어 및/또는 하드웨어 플러그가능 모듈로서 구현될 수 있다. 예를 들어, 고속 인터페이스 및 메모리(예를 들어, 비휘발성 또는 플래쉬 메모리)를 갖는 카드가 프로세서 서브시스템에의 적당한 인터페이스를 위해 또한 솔버 상태 처리를 위해 이용될 수 있다. 다른 대안으로서 또는 이들과 조합하여, 개시된 발명에 따라 솔버 상태를 처리하는 소프트웨어 모듈이 설치될 수 있다.

[0112] 도 13은 본 발명에 따른 솔버 상태 처리를 개별적인 컴퓨팅 장치들에 걸친 솔버 상태 처리에 적용하는 시스템(1300)을 나타낸 도면이다. 여기에서, 제1 컴퓨팅 시스템(1302)은 프로그램 및 데이터 처리를 수행하는 시스템 프로세서(1304)(및 단일 코어(1306))를 갖는 단일-프로세서 시스템이다. 제2 컴퓨팅 시스템(1308)은 프로그램 및 데이터 처리를 수행하는 멀티-프로세서 서브시스템(1310)(및 2개의 프로세서(1312, 1314))을 갖는 멀티-프로세서 시스템이다. 시스템(1302, 1308)은 네트워크(1316)(또는 적당한 고속 인터페이스)를 통해 통신하게 또한 솔버 상태 처리를 위해 도 12의 상태 시스템(1226)과 통신하게 배치되어 있다. 양 시스템(1302, 1308)에 걸쳐 쓰레드가 처리되는 경우, 도 12와 관련하여 상기한 설명과 유사하게 양 시스템으로부터 솔버 상태가 처리될 수 있다.

[0113] 도 14는 큰 탐색 공간에서 문제 솔버로서 적용하기 위해 TMS(truth maintenance system)(1402) 및 추론 엔진(1404)(학습 및 추론을 사용함)을 이용하는 CSP 솔버 시스템(1400)을 나타낸 도면이다. 엔진(1404)은 대안을 탐색하고, 선택을 하며, 정확성을 위해 그 선택들을 분석한다. 충돌이 발생하는 경우, TMS는 충돌의 제거를 용이하게 해주고, 그에 따라 장래의 사용을 위해 그의 지식 베이스를 갱신한다.

[0114] 본 출원에서 사용되는 바와 같이, 용어 "컴포넌트" 및 "시스템"은 컴퓨터-관련 개체(하드웨어, 하드웨어와 소프트웨어의 조합, 소프트웨어, 또는 실행 중인 소프트웨어)를 말하기 위한 것이다. 예를 들어, 컴포넌트는 프로세서 상에서 실행 중인 프로세스, 프로세서, 하드 디스크 드라이브, (광 및/또는 자기 저장 매체의) 다수의 저장 장치 드라이브, 객체, 실행 파일, 실행 쓰레드, 프로그램, 및/또는 컴퓨터일 수 있지만, 이에 한정되지 않는다. 예시로서, 서버 상에서 실행 중인 애플리케이션 및 서버 둘다가 컴포넌트일 수 있다. 프로세스 및/또는

실행 쓰레드 내에 하나 이상의 컴포넌트가 존재할 수 있고, 컴포넌트는 하나의 컴퓨터 상에 로컬화되어 있고 및/또는 2개 이상의 컴퓨터 간에 분산되어 있을 수 있다.

[0115] 이제 도 15를 참조하면, 개시된 병렬 솔버 상태 아키텍처를 이용할 수 있는 컴퓨팅 시스템(1500)의 블록도가 도시되어 있다. 이 아키텍처의 다양한 측면들에 대한 부가적인 상황을 제공하기 위해, 도 15 및 이하의 설명은 본 발명의 다양한 측면들이 구현될 수 있는 적당한 컴퓨팅 시스템(1500)의 간략한 전반적인 설명을 제공하기 위한 것이다. 이상의 설명이 일반적으로 하나 이상의 컴퓨터 상에서 실행될 수 있는 컴퓨터-실행가능 명령어와 관련되어 있지만, 당업자라면 본 발명이 또한 다른 프로그램 모듈들과 결합하여 및/또는 하드웨어와 소프트웨어의 조합으로서 구현될 수 있다는 것을 잘 알 것이다.

[0116] 일반적으로, 프로그램 모듈은 특정의 작업을 수행하거나 특정의 추상 데이터 유형을 구현하는 루틴, 프로그램, 컴포넌트, 데이터 구조, 기타 등등을 포함한다. 게다가, 당업자라면 본 발명의 방법들이 다른 컴퓨터 시스템 구성(단일-프로세서 또는 멀티프로세서 컴퓨터 시스템, 미니컴퓨터, 메인프레임 컴퓨터는 물론 퍼스널 컴퓨터, 핸드헬드 컴퓨팅 장치, 마이크로프로세서-기반 또는 프로그램가능 가전 제품, 기타 등등을 포함하며, 이들 각각은 하나 이상의 관련 장치에 연결되어 동작할 수 있음)으로 실시될 수 있다는 것을 잘 알 것이다.

[0117] 본 발명의 예시된 측면들은 또한 어떤 작업들이 통신 네트워크를 통해 연결되어 있는 원격 처리 장치들에 의해 수행되는 분산 컴퓨팅 환경에서도 실시될 수 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈들은 로컬 및 원격 메모리 저장 장치 둘다에 위치될 수 있다.

[0118] 컴퓨터는 통상적으로 각종의 컴퓨터-판독가능 매체를 포함한다. 컴퓨터-판독가능 매체는 컴퓨터에 의해 액세스될 수 있는 이용가능한 매체라면 어느 것이라도 될 수 있고 휘발성 및 비휘발성 매체, 이동식 및 비이동식 매체를 포함한다. 제한이 아닌 예로서, 컴퓨터-판독가능 매체는 컴퓨터 저장 매체 및 통신 매체를 포함할 수 있다. 컴퓨터 저장 매체는 컴퓨터-판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터 등의 정보를 저장하는 임의의 방법 또는 기술로 구현되는 휘발성 및 비휘발성, 이동식 및 비이동식 매체 둘다를 포함한다. 컴퓨터 저장 매체는 RAM, ROM, EEPROM, 플래쉬 메모리 또는 기타 메모리 기술, CD-ROM, DVD(digital video disk) 또는 기타 광 디스크 저장 장치, 자기 카세트, 자기 테이프, 자기 디스크 저장 장치 또는 기타 자기 저장 장치, 또는 원하는 정보를 저장하는 데 사용될 수 있고 컴퓨터에 의해 액세스될 수 있는 임의의 다른 매체를 포함하지만, 이에 한정되지 않는다.

[0119] 다시 도 15를 참조하면, 여러가지 측면들을 구현하는 예시적인 컴퓨팅 시스템(1500)은 컴퓨터(1502)를 포함하며, 이 컴퓨터(1502)는 처리 장치(1504), 시스템 메모리(1506) 및 시스템 버스(1508)를 포함한다. 시스템 버스(1508)는 시스템 메모리(1506)(이에 한정되지 않음)를 비롯한 시스템 컴포넌트들의 처리 장치(1504)에 대한 인터페이스를 제공한다. 처리 장치(1504)는 다양한 상용 프로세서들 중 어느 것이라도 될 수 있다. 듀얼 마이크로프로세서 및 기타 멀티-프로세서 아키텍처도 처리 장치(1504)로서 이용될 수 있다.

[0120] 시스템 버스(1508)는 메모리 버스(메모리 컨트롤러를 갖거나 갖지 않음), 주변 장치 버스, 및 각종의 상용 버스 아키텍처 중 임의의 것을 사용하는 로컬 버스에 추가적으로 상호연결될 수 있는 몇가지 유형의 버스 구조 중 어느 것이라도 될 수 있다. 시스템 메모리(1506)는 판독 전용 메모리(ROM)(1510) 및 랜덤 액세스 메모리(RAM)(1512)를 포함한다. 기본 입/출력 시스템(BIOS)은 ROM, EPROM, EEPROM 등의 비휘발성 메모리(1510)에 저장되며, 이 BIOS는 시동 중과 같은 때에 컴퓨터(1502) 내의 구성요소들 간의 정보 전송을 돕는 기본적인 루틴을 포함하고 있다. RAM(1512)은 또한 데이터를 캐싱하기 위한 정적 RAM 등의 고속 RAM을 포함할 수 있다.

[0121] 컴퓨터(1502)는 또한 내장형 하드 디스크 드라이브(HDD)(1514)(예를 들어, EIDE, SATA)(이 내장형 하드 디스크 드라이브(1514)는 적당한 섀시(도시 생략)에 넣어 외장형으로 구성될 수도 있음), 자기 플로피 디스크 드라이브(FDD)(1516)(예를 들어, 이동식 디스켓(1518)으로부터 판독을 하거나 그에 기록을 하기 위한 것임), 및 광 디스크 드라이브(1520)(예를 들어, CD-ROM 디스크(1522)를 판독하거나, DVD 등의 기타 대용량 광 매체로부터 판독을 하거나 그에 기록을 하기 위한 것임)를 포함하고 있다. 하드 디스크 드라이브(1514), 자기 디스크 드라이브(1516) 및 광 디스크 드라이브(1520)는 각각 하드 디스크 드라이브 인터페이스(1524), 자기 디스크 드라이브 인터페이스(1526) 및 광 드라이브 인터페이스(1528)에 의해 시스템 버스(1508)에 연결될 수 있다. 외장형 드라이브 구현을 위한 인터페이스(1524)는 USB(Universal Serial Bus) 및 IEEE 1394 인터페이스 기술 중 적어도 하나 또는 그 둘다를 포함한다. 다른 외장형 드라이브 접속 기술이 본 발명에서 생각되고 있다.

[0122] 이들 드라이브 및 그의 관련 컴퓨터-판독가능 매체는 데이터, 데이터 구조, 컴퓨터-실행가능 명령어, 기타 등등의 비휘발성 저장을 제공한다. 컴퓨터(1502)에서, 드라이브 및 매체는 임의의 데이터를 적당한 디지털 형태로

저장하는 것에 대응하고 있다. 컴퓨터-판독가능 매체에 대한 이상의 설명이 HDD, 이동식 자기 디스켓, 및 CD 또는 DVD 등의 이동식 광 매체를 언급하고 있지만, 당업자라면 컴퓨터에 의해 판독가능한 다른 유형의 매체(zip 드라이브, 자기 카세트, 플래쉬 메모리 카드, 카트리지, 기타 등등)도 역시 예시적인 운영 환경에서 사용될 수 있다는 것과 또한 임의의 이러한 매체가 본 발명의 방법들을 수행하는 컴퓨터-실행가능 명령어를 포함하고 있을 수 있다는 것을 잘 알 것이다.

[0123] 운영 체제(1530), 하나 이상의 애플리케이션(1532)(예를 들어, 전술한 잠금없는 CSP 솔버 처리 시스템), 기타 프로그램 모듈(1534) 및 프로그램 데이터(1536)를 비롯한 다수의 프로그램 모듈이 이들 드라이브 및 RAM(1512)에 저장될 수 있다. 운영 체제, 애플리케이션, 모듈 및/또는 데이터의 전부 또는 그의 일부분은 또한 RAM(1512)에 캐싱될 수 있다. 본 발명이 다양한 상용 운영 체제 또는 운영 체제들의 조합으로 구현될 수 있다는 것을 잘 알 것이다.

[0124] 사용자는 하나 이상의 유선/무선 입력 장치(예를 들어, 키보드(1538), 및 마우스(1540) 등의 포인팅 장치)를 통해 컴퓨터(1502)에 명령 및 정보를 입력할 수 있다. 기타 입력 장치(도시 생략)로는 마이크, IR 리모콘, 조이스틱, 게임 패드, 스타일러스 펜, 터치 스크린, 기타 등등이 있을 수 있다. 이들 및 기타 입력 장치는 종종 시스템 버스(1508)에 연결되어 있는 입력 장치 인터페이스(1542)를 통해 처리 장치(1504)에 접속되어 있지만, 병렬 포트, IEEE 1394 직렬 포트, 게임 포트, USB 포트, IR 인터페이스, 기타 등등의 다른 인터페이스에 의해 접속될 수 있다.

[0125] 모니터(1544) 또는 기타 유형의 디스플레이 장치도 비디오 어댑터(1546) 등의 인터페이스를 통해 시스템 버스(1508)에 접속되어 있다. 모니터(1544)에 부가하여, 컴퓨터는 통상적으로 스피커, 프린터, 기타 등등의 다른 주변 출력 장치(도시 생략)를 포함하고 있다.

[0126] 컴퓨터(1502)는 원격 컴퓨터(들)(1548) 등의 하나 이상의 원격 컴퓨터로의 유선 및/또는 무선 통신을 통한 논리적 접속을 사용하여 네트워크화된 환경에서 동작할 수 있다. 원격 컴퓨터(들)(1548)는 워크스테이션, 서버 컴퓨터, 라우터, 퍼스널 컴퓨터, 휴대용 컴퓨터, 마이크로프로세서-기반 오락 기기, 피어 장치 또는 기타 통상의 네트워크 노드일 수 있으며, 통상적으로 컴퓨터(1502)와 관련하여 기술된 구성요소들 중 다수 또는 그 전부를 포함하지만, 간략함을 위해 메모리/저장 장치(1550)만이 도시되어 있다. 도시된 논리적 접속은 근거리 통신망(LAN)(1552) 및/또는 더 큰 네트워크(예를 들어, 원거리 통신망(WAN)(1544))에 유선/무선 접속을 포함한다. 이러한 LAN 및 WAN 네트워킹 환경은 사무실 및 회사에서 흔한 것이며, 인트라넷 등의 전사적 컴퓨터 네트워크를 용이하게 해주고, 이들 모두는 글로벌 통신 네트워크(예를 들어, 인터넷)에 접속될 수 있다.

[0127] LAN 네트워킹 환경에서 사용될 때, 컴퓨터(1502)는 유선 및/또는 무선 통신 네트워크 인터페이스 또는 어댑터(1556)를 통해 로컬 네트워크(1552)에 접속되어 있다. 어댑터(1556)는 LAN(1552)(무선 어댑터(1556)와 통신하기 위해 그 위에 배치된 무선 액세스 포인트를 포함할 수도 있음)로의 유선 또는 무선 통신을 용이하게 해줄 수 있다.

[0128] WAN 네트워킹 환경에서 사용될 때, 컴퓨터(1502)는 모뎀(1558)을 포함할 수 있거나, WAN(1554) 상의 통신 서버에 연결되어 있거나, 인터넷을 통하는 등 WAN(1554)을 통해 통신을 설정하는 기타 수단을 갖는다. 내장형 또는 외장형, 유선 또는 무선 장치일 수 있는 모뎀(1558)은 직렬 포트 인터페이스(1542)를 통해 시스템 버스(1508)에 연결되어 있다. 네트워크화된 환경에서, 컴퓨터(1502)와 관련하여 설명된 프로그램 모듈 또는 그의 일부분이 원격 메모리/저장 장치(1550)에 저장되어 있을 수 있다. 도시된 네트워크 접속이 예시적인 것이며 컴퓨터들 간에 통신 링크를 설정하는 기타 수단이 사용될 수 있다는 것을 잘 알 것이다.

[0129] 컴퓨터(1502)는 무선 통신에 배치되어 동작하는 임의의 무선 장치 또는 개체[예를 들어, 프린터, 스캐너, 데스크톱 및/또는 휴대용 컴퓨터, PDA(portable data assistant), 통신 위성, 무선 검출가능 태그와 연관된 임의의 장비 또는 장소(예를 들어, 키오스크, 신문 가판대, 휴게실) 및 전화]와 통신을 할 수 있다. 이것은 적어도 Wi-Fi 및 블루투스™ 무선 기술을 포함한다. 따라서, 통신은 종래의 네트워크에서와 같이 미리 정의된 구조일 수 있거나, 간단히 적어도 2개의 장치들 간의 애드혹 통신일 수 있다.

[0130] 이제 도 16을 참조하면, 개시된 발명에 따른 병렬 솔버 처리를 이용할 수 있는 예시적인 컴퓨팅 환경(1600)의 개략 블록도가 도시되어 있다. 시스템(1600)은 하나 이상의 클라이언트(들)(1602)를 포함하고 있다. 클라이언트(들)(1602)는 하드웨어 및/또는 소프트웨어(예를 들어, 쓰레드, 프로세스, 컴퓨팅 장치)일 수 있다. 클라이언트(들)(1602)는, 예를 들어, 본 발명을 이용함으로써, 쿠키(들) 및/또는 관련 컨텍스트 정보를 가지고 있을 수 있다.

[0131] 시스템(1600)은 또한 하나 이상의 서버(들)(1604)를 포함하고 있다. 서버(들)(1604)도 역시 하드웨어 및/또는 소프트웨어(예를 들어, 쓰레드, 프로세스, 컴퓨팅 장치)일 수 있다. 서버(1604)는, 예를 들어, 본 아키텍처를 이용하여 변환을 수행하는 쓰레드를 가지고 있을 수 있다. 클라이언트(1602)와 서버(1604) 간의 한가지 가능한 통신은 병렬 솔버 상태 처리를 지원하기 위해 하나 이상의 컴퓨터 프로세스 간에 전송되도록 구성된 데이터 패킷의 형태로 되어 있을 수 있다. 이 데이터 패킷은, 예를 들어, 쿠키 및/또는 관련 컨텍스트 정보를 포함하고 있을 수 있다. 시스템(1600)은 클라이언트(들)(1602)와 서버(들)(1604) 간의 통신을 용이하게 해주는 데 사용될 수 있는 통신 프레임워크(1606)(예를 들어, 인터넷 등의 글로벌 통신 네트워크)를 포함하고 있다.

[0132] 통신은 유선(광 섬유를 포함함) 및/또는 무선 기술에 의해 용이하게 될 수 있다. 클라이언트(들)(1602)는 클라이언트(들)(1602)에 로컬인 정보(예를 들어, 쿠키 및/또는 관련 컨텍스트 정보)를 저장하는 데 이용될 수 있는 하나 이상의 클라이언트 데이터 저장소(들)(1608)에 연결되어 동작한다. 이와 유사하게, 서버(들)(1604)는 서버(1604)에 로컬인 정보를 저장하는 데 이용될 수 있는 하나 이상의 서버 데이터 저장소(들)(1610)에 연결되어 동작한다.

산업상 이용 가능성

[0133] 이상에서, 본 발명의 예들에 대해 기술하였다. 물론, 컴포넌트들 및/또는 방법들의 모든 생각가능한 조합을 기술할 수는 없지만, 당업자라면 많은 추가의 조합 및 치환이 가능하다는 것을 잘 알 것이다. 그에 따라, 본 발명은 첨부된 청구항들의 정신 및 범위 내에 속하는 이러한 변경, 수정 및 변형 모두를 포함하는 것으로 보아야 한다. 게다가, 용어 "포함한다"가 상세한 설명 또는 청구항에서 사용되는 한, 이러한 용어는 용어 "포함하는"이 청구항에서 이행구로 이용될 때 해석되는 것과 유사한 방식으로 포함적인 것으로 보아야 한다.

도면의 간단한 설명

[0010] 도 1은 병렬 구현에 따라 솔버 처리를 용이하게 해주는 시스템을 나타낸 도면.

[0011] 도 2는 본 발명에 따른 솔버 상태 처리 방법을 나타낸 도면.

[0012] 도 3은 대답 할당(answer assignment)에 대한 추측을 추리하기 위해 학습 및 추론을 이용하는 대안의 병렬 솔버 시스템을 나타낸 도면.

[0013] 도 4는 예시적인 제약조건 전파 프로세스로부터 도출되는 (L,K)-연역 그래프(deduction graph)를 나타낸 도면.

[0014] 도 5는 도 4의 지원 그래프로부터의 충돌하는 제약조건 출력을 처리하는 새로운 그래프를 나타낸 도면.

[0015] 도 6은 2개의 지원 그래프를 작성하여 병합하는 방법을 나타낸 도면.

[0016] 도 7은 쌍으로 병합하는 프로세스(pairwise merging process)에 대한 2개의 입력 지원 그래프를 삭감하는 방법을 나타낸 도면.

[0017] 도 8은 병합된 그래프의 충돌을 처리하는 방법을 나타낸 도면.

[0018] 도 9는 지원 그래프를 삭감 및 병합하여 순차 SAT 솔버에 대한 최종적인 병합된 충돌없는 그래프를 산출하는 방법을 나타낸 도면.

[0019] 도 10은 지원 그래프를 삭감 및 병합하여 병렬 SAT 솔버에 대한 최종적인 병합된 충돌없는 그래프를 산출하는 방법을 나타낸 도면.

[0020] 도 11은 본 발명의 솔버 상태 처리를 멀티-코어 처리 시스템에 적용하는 시스템을 나타낸 도면.

[0021] 도 12는 본 발명에 따른 솔버 상태 처리를 멀티-코어 멀티-프로세서 시스템에 적용하는 시스템을 나타낸 도면.

[0022] 도 13은 본 발명에 따른 솔버 상태 처리를 서로 다른 컴퓨팅 시스템들에 걸친 솔버 상태 처리에 적용하는 시스템을 나타낸 도면.

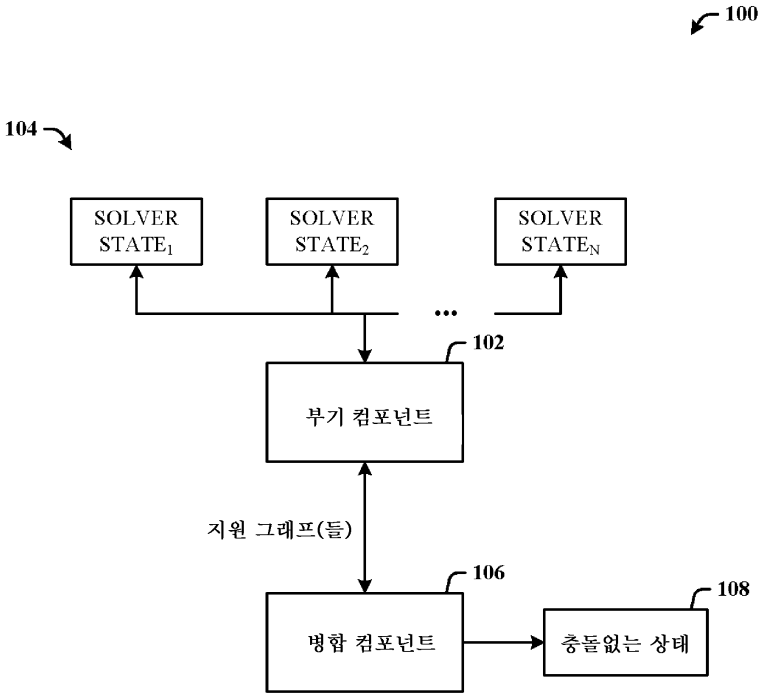
[0023] 도 14는 큰 탐색 공간에서 문제 솔버로서 적용하기 위해 TMS(truth maintenance system) 및 추론 엔진을 이용하는 CSP 솔버 시스템을 나타낸 도면.

[0024] 도 15는 개시된 병렬 솔버 상태 아키텍처를 이용할 수 있는 컴퓨팅 시스템의 블록도.

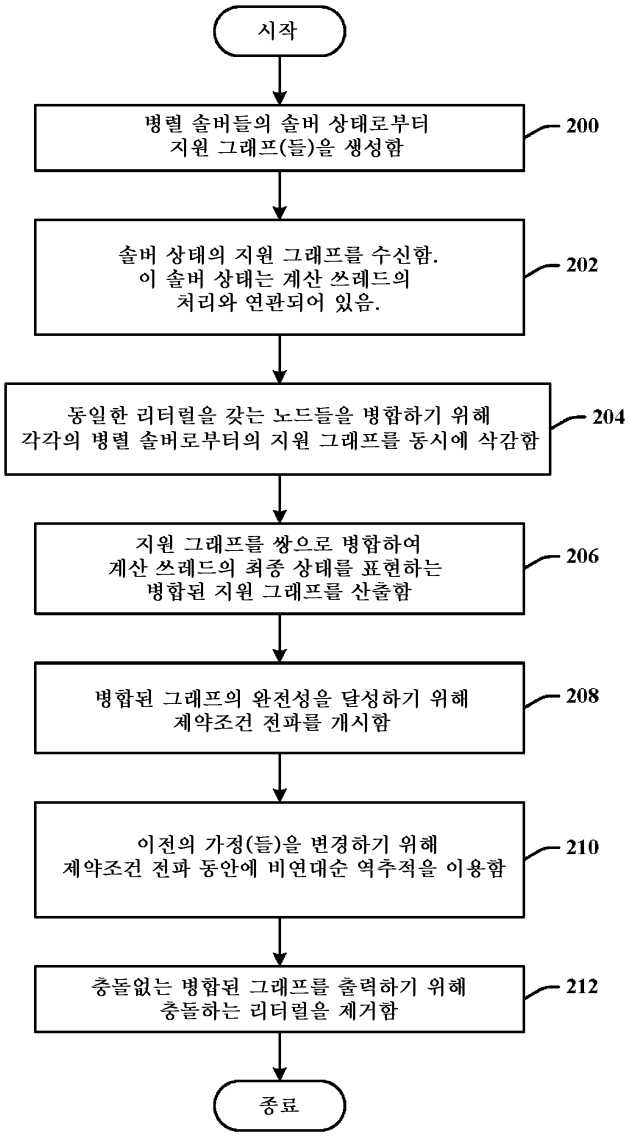
[0025] 도 16은 개시된 발명에 따른 병렬 솔버 처리를 이용할 수 있는 예시적인 컴퓨팅 환경의 개략 블록도.

도면

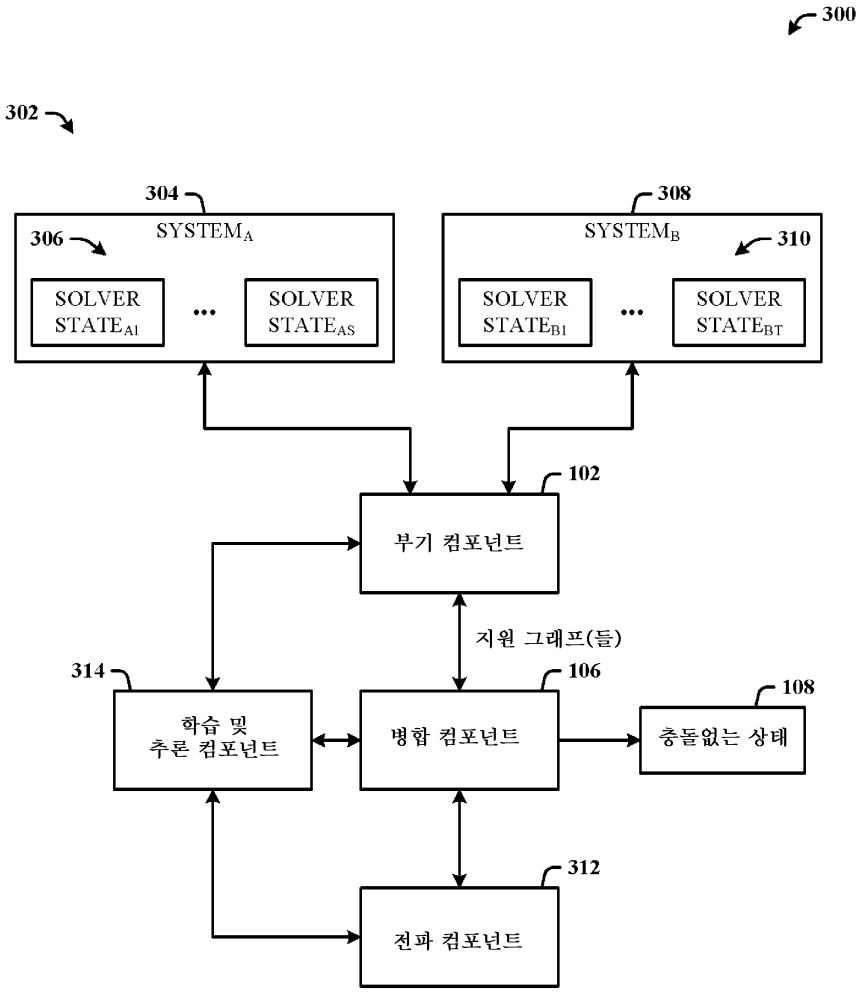
도면1



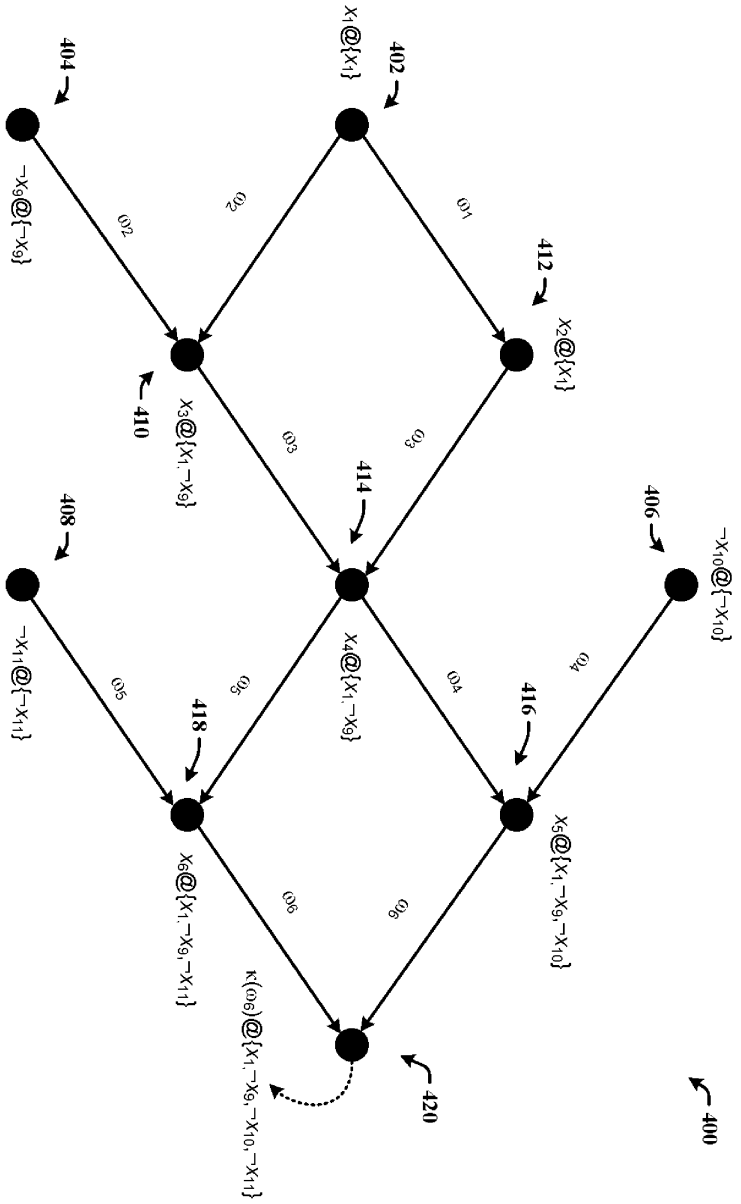
도면2


시작
병렬 솔버들의 솔버 상태로부터 지원 그래프(들)을 생성함
200
솔버 상태의 지원 그래프를 수신함. 이 솔버 상태는 계산 쓰레드의 처리와 연관되어 있음.
202
동일한 리터럴을 갖는 노드들을 병합하기 위해 각각의 병렬 솔버로부터의 지원 그래프를 동시에 삭감함
204
지원 그래프를 쌍으로 병합하여 계산 쓰레드의 최종 상태를 표현하는 병합된 지원 그래프를 산출함
206
병합된 그래프의 완전성을 달성하기 위해 제약조건 전파를 개시함
208
이전의 가정(들)을 번경하기 위해 제약조건 전파 동안에 비연대순 역추적을 이용함
210
충돌없는 병합된 그래프를 출력하기 위해 충돌하는 리터럴을 제거함
212
종료

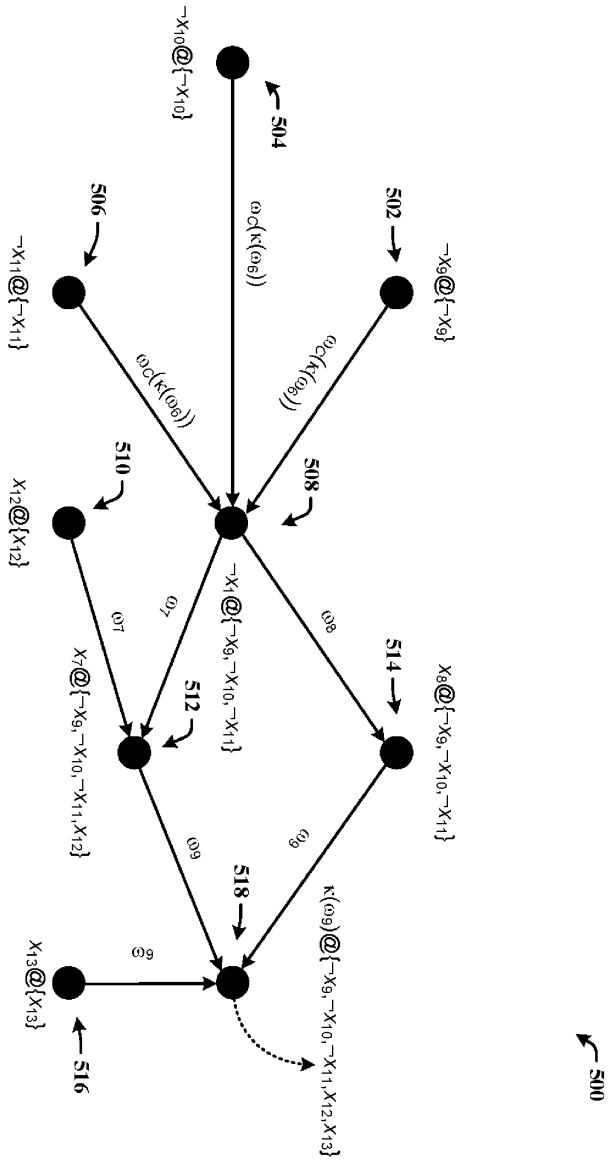
도면3


300
302
304
SYSTEM$_A$
306
SOLVER STATE$_{A1}$
...
SOLVER STATE$_{AS}$
308
SYSTEM$_B$
310
SOLVER STATE$_{B1}$
...
SOLVER STATE$_{BT}$
102
부기 컴포넌트
지원 그래프(들)
314
학습 및 추론 컴포넌트
106
병합 컴포넌트
108
충돌없는 상태
312
전파 컴포넌트

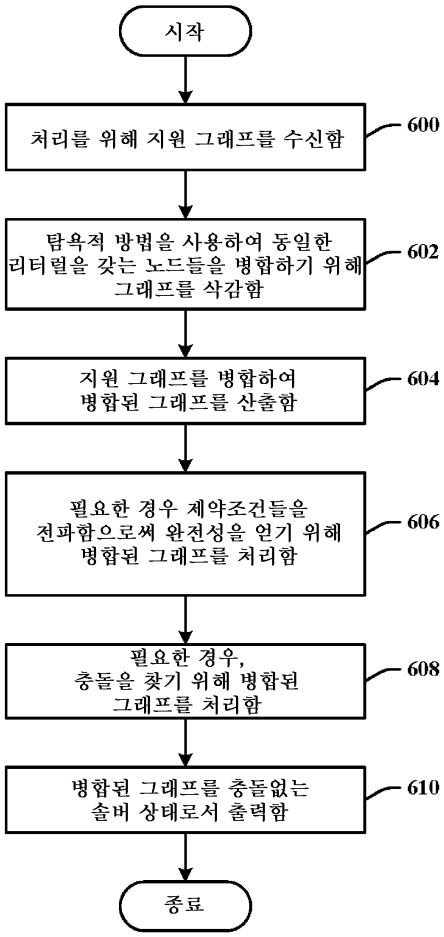
도면4



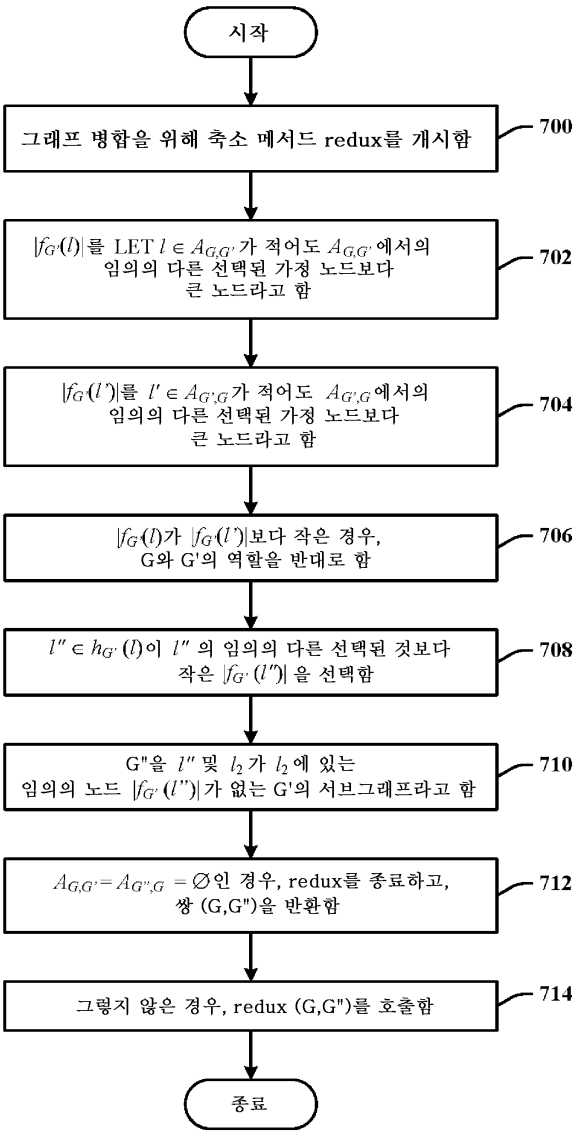
도면5


¬x10@{¬x10}
504
506
¬x11@{¬x11}
502
¬x9@{¬x9}
ω6c(κ(ω6))
ω5c(κ(ω5))
ω8c(κ(ω8))
510
x12@{x12}
508
¬x1@{¬x9,¬x10,¬x11}
ω7
ω8
514
x8@{¬x9,¬x10,¬x11}
x7@{¬x9,¬x10,¬x11,x12}
512
ω9
ω9
518
ω6
x13@{x13}
κ(ω9)@{¬x9,¬x10,¬x11,x12,x13}
516
500

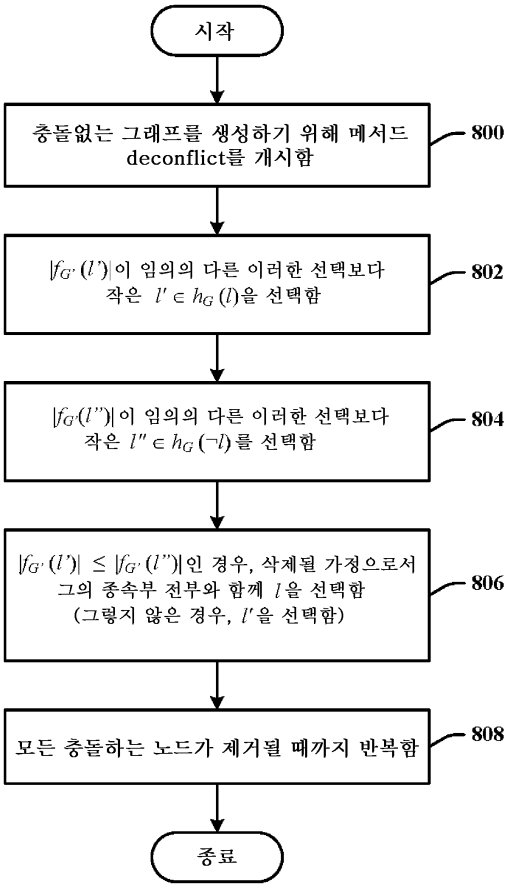
도면6


시작
처리를 위해 지원 그래프를 수신함
600
탐욕적 방법을 사용하여 동일한 리터럴을 갖는 노드들을 병합하기 위해 그래프를 삭감함
602
지원 그래프를 병합하여 병합된 그래프를 산출함
604
필요한 경우 제약조건들을 전파함으로써 완전성을 얻기 위해 병합된 그래프를 처리함
606
필요한 경우, 충돌을 찾기 위해 병합된 그래프를 처리함
608
병합된 그래프를 충돌없는 솔버 상태로서 출력함
610
종료

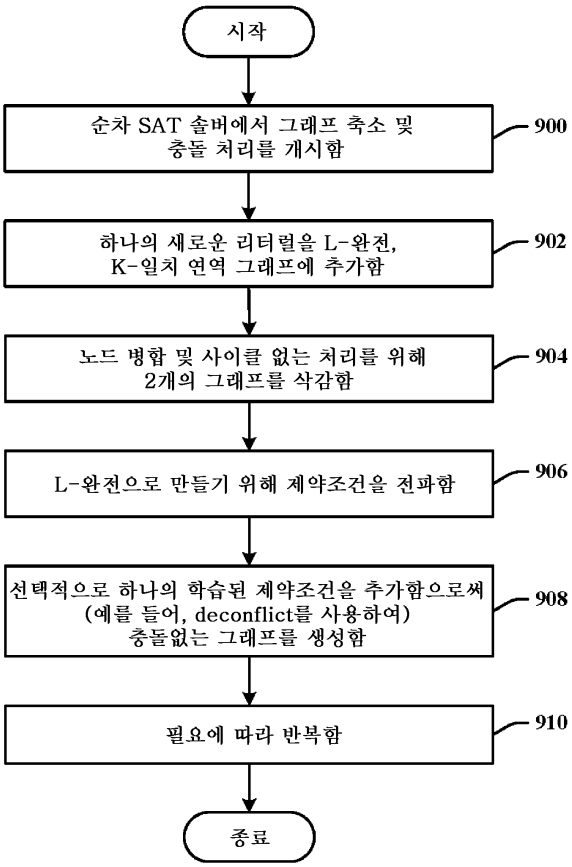
도면7


시작
그래프 병합을 위해 축소 메서드 redux를 개시함
700
$|f_{G'}(l)|$를 LET $l \in A_{G,G'}$가 적어도 $A_{G,G'}$에서의 임의의 다른 선택된 가정 노드보다 큰 노드라고 함
702
$|f_G(l')|$를 $l' \in A_{G',G}$가 적어도 $A_{G',G}$에서의 임의의 다른 선택된 가정 노드보다 큰 노드라고 함
704
$|f_G(l)|$가 $|f_G(l')|$보다 작은 경우, G와 G'의 역할을 반대로 함
706
$l'' \in h_{G'}(l)$이 l''의 임의의 다른 선택된 것보다 작은 $|f_{G'}(l'')|$을 선택함
708
G"을 l'' 및 l_2가 l_2에 있는 임의의 노드 $|f_{G'}(l'')|$가 없는 G'의 서브그래프라고 함
710
$A_{G,G'} = A_{G'',G} = \varnothing$인 경우, redux를 종료하고, 쌍 (G,G")을 반환함
712
그렇지 않은 경우, redux (G,G")를 호출함
714
종료

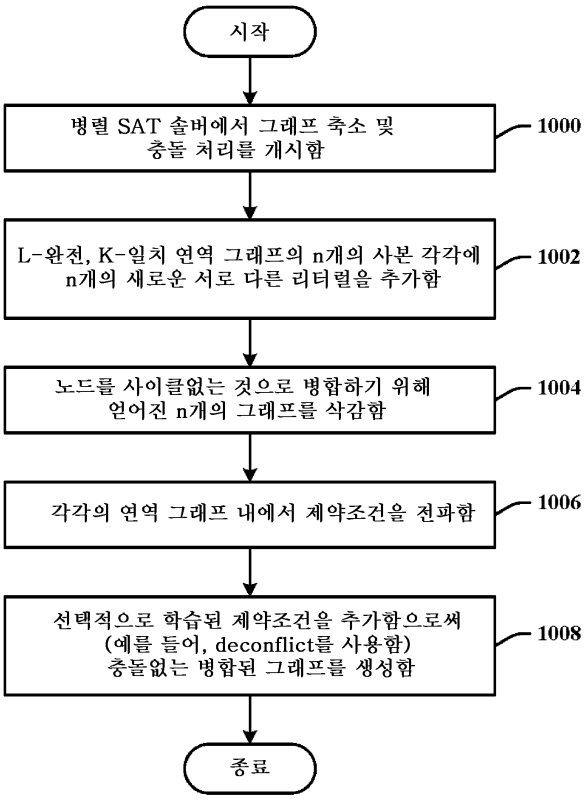
도면8


시작
충돌없는 그래프를 생성하기 위해 메서드 deconflict를 개시함
800
$|f_{G'}(l')|$이 임의의 다른 이러한 선택보다 작은 $l' \in h_G(l)$을 선택함
802
$|f_{G'}(l'')|$이 임의의 다른 이러한 선택보다 작은 $l'' \in h_G(\neg l)$를 선택함
804
$|f_{G'}(l')| \leq |f_{G'}(l'')|$인 경우, 삭제될 가정으로서 그의 종속부 전부와 함께 l을 선택함 (그렇지 않은 경우, l'을 선택함)
806
모든 충돌하는 노드가 제거될 때까지 반복함
808
종료

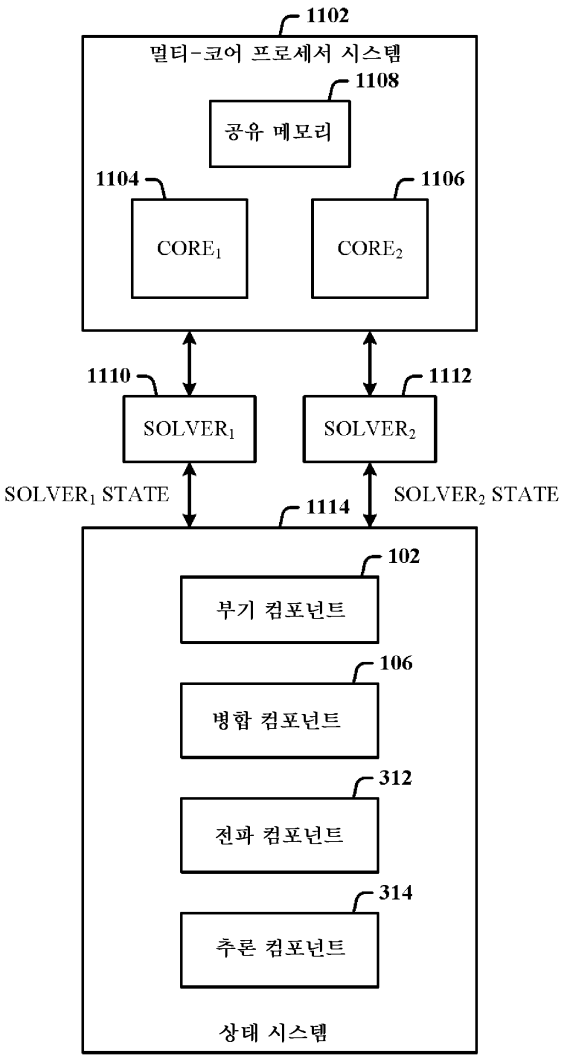
도면9


시작
순차 SAT 솔버에서 그래프 축소 및 충돌 처리를 개시함
900
하나의 새로운 리터럴을 L-완전, K-일치 연역 그래프에 추가함
902
노드 병합 및 사이클 없는 처리를 위해 2개의 그래프를 삭감함
904
L-완전으로 만들기 위해 제약조건을 전파함
906
선택적으로 하나의 학습된 제약조건을 추가함으로써 (예를 들어, deconflict를 사용하여) 충돌없는 그래프를 생성함
908
필요에 따라 반복함
910
종료

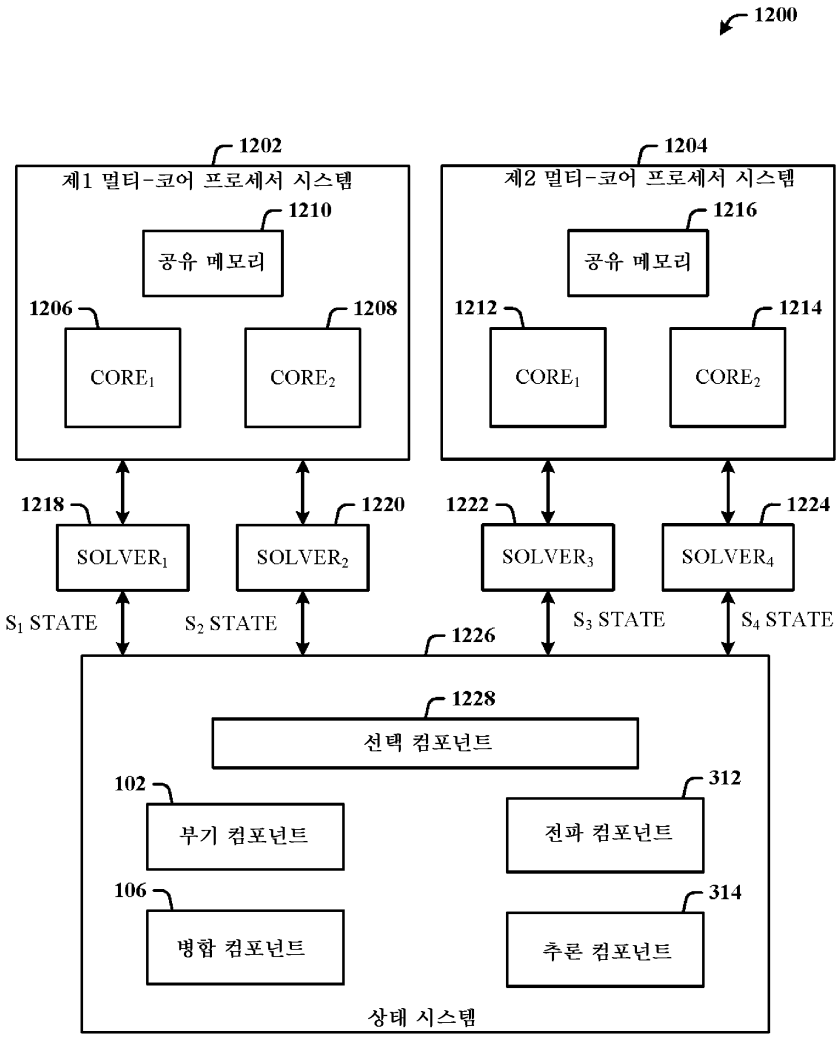
도면10


시작
병렬 SAT 솔버에서 그래프 축소 및 충돌 처리를 개시함
1000
L-완전, K-일치 연역 그래프의 n개의 사본 각각에 n개의 새로운 서로 다른 리터럴을 추가함
1002
노드를 사이클없는 것으로 병합하기 위해 얻어진 n개의 그래프를 삭감함
1004
각각의 연역 그래프 내에서 제약조건을 전파함
1006
선택적으로 학습된 제약조건을 추가함으로써 (예를 들어, deconflict를 사용함) 충돌없는 병합된 그래프를 생성함
1008
종료

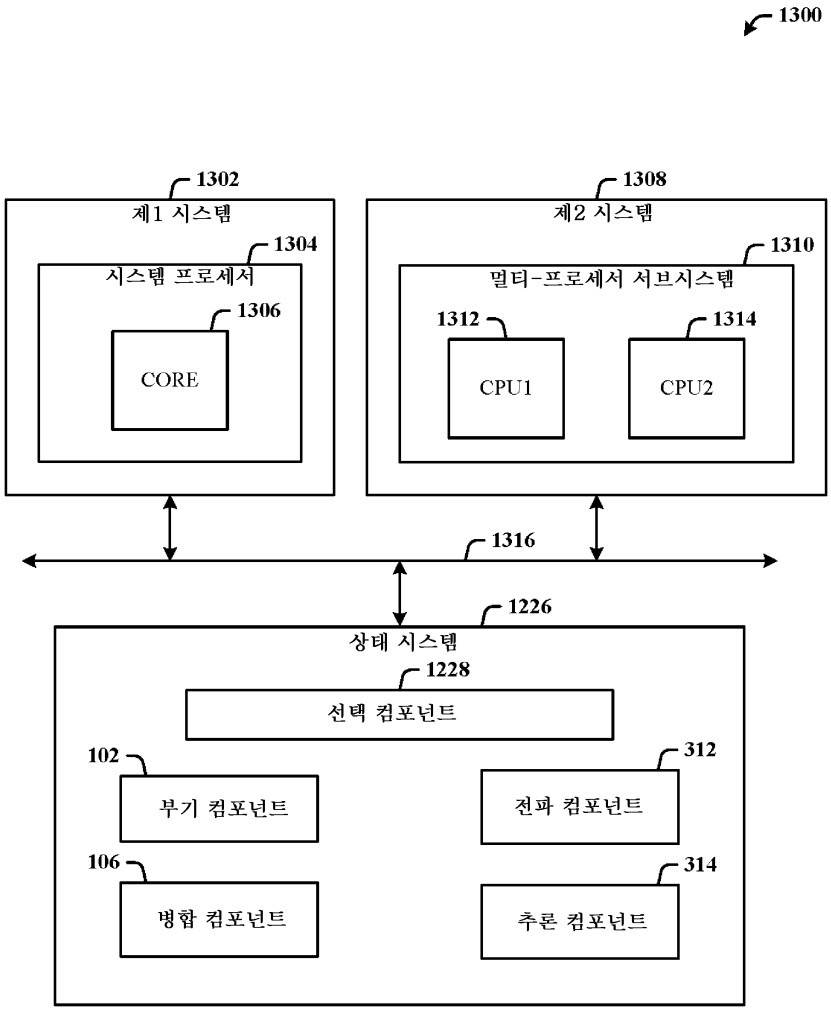
도면11


1102
멀티-코어 프로세서 시스템
1108
공유 메모리
1104
CORE1
1106
CORE2
1110
SOLVER1
1112
SOLVER2
SOLVER1 STATE
SOLVER2 STATE
1114
102
부기 컴포넌트
106
병합 컴포넌트
312
전파 컴포넌트
314
추론 컴포넌트
상태 시스템

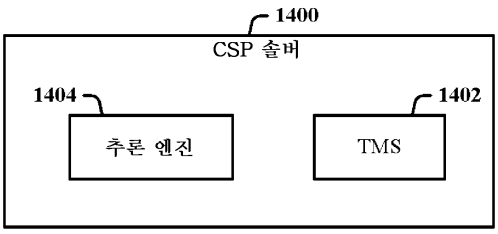
도면12


1200
1202
제1 멀티-코어 프로세서 시스템
1210
공유 메모리
1206
CORE1
1208
CORE2
1204
제2 멀티-코어 프로세서 시스템
1216
공유 메모리
1212
CORE1
1214
CORE2
1218
SOLVER1
1220
SOLVER2
1222
SOLVER3
1224
SOLVER4
S1 STATE
S2 STATE
S3 STATE
S4 STATE
1226
1228
선택 컴포넌트
102
부기 컴포넌트
312
전파 컴포넌트
106
병합 컴포넌트
314
추론 컴포넌트
상태 시스템

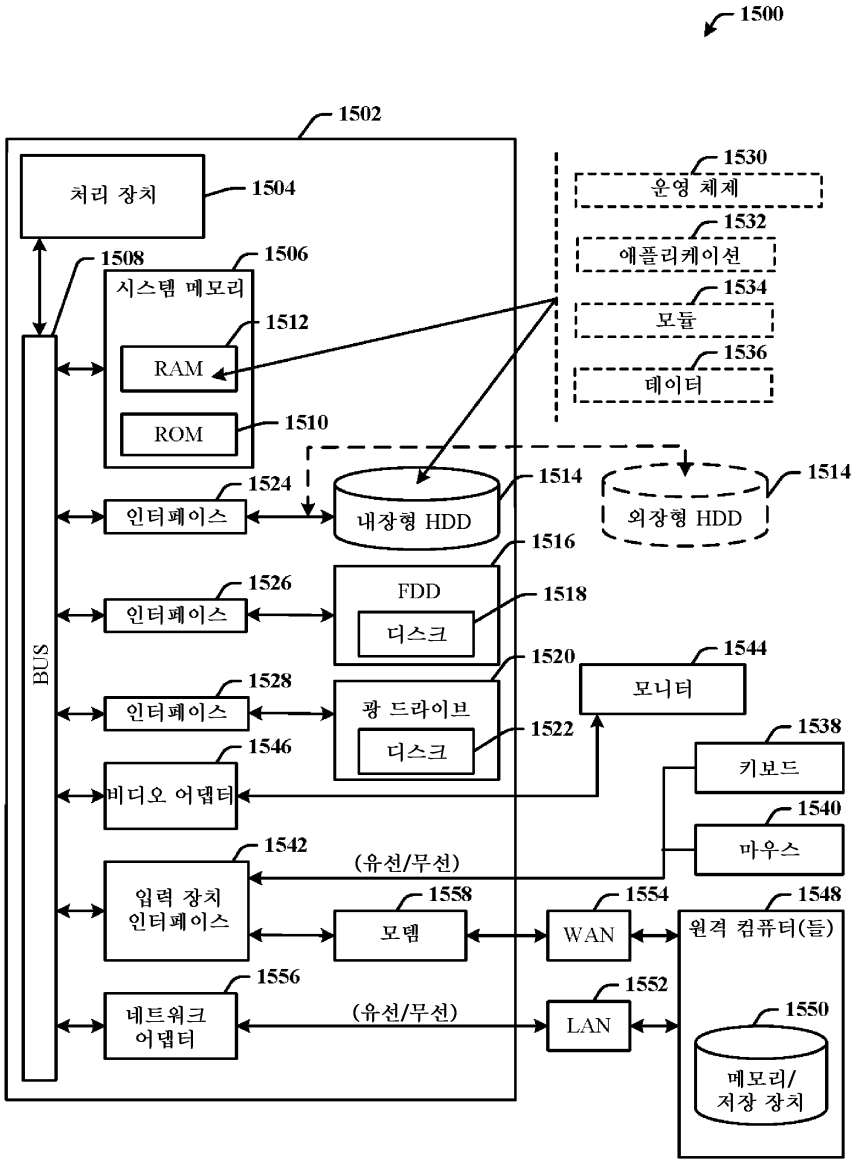
도면13


1300
1302
제1 시스템
1304
시스템 프로세서
1306
CORE
1308
제2 시스템
1310
멀티-프로세서 서브시스템
1312
CPU1
1314
CPU2
1316
1226
상태 시스템
1228
선택 컴포넌트
102
부기 컴포넌트
312
전파 컴포넌트
106
병합 컴포넌트
314
추론 컴포넌트

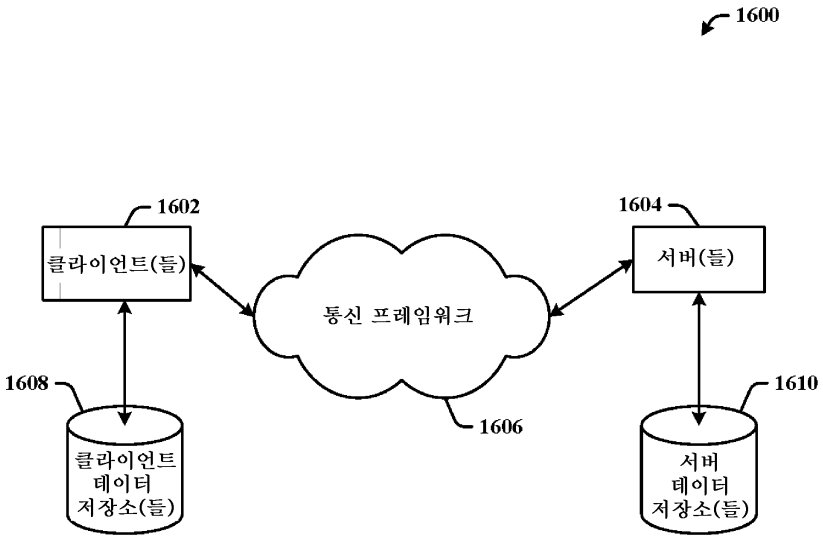
도면14


1400
CSP 솔버
1404
추론 엔진
1402
TMS

도면15


1500
1502
처리 장치
1504
1508
1506
시스템 메모리
1512
RAM
ROM
1510
BUS
1530
운영 체제
1532
애플리케이션
1534
모듈
1536
데이터
1524
인터페이스
내장형 HDD
1514
외장형 HDD
1514
1516
1526
인터페이스
FDD
디스크
1518
1528
인터페이스
광 드라이브
디스크
1520
1522
1544
모니터
1538
키보드
1540
마우스
1546
비디오 어댑터
1542
입력 장치
인터페이스
(유선/무선)
1558
모뎀
1554
WAN
1548
원격 컴퓨터(들)
1556
네트워크
어댑터
(유선/무선)
1552
LAN
1550
메모리/
저장 장치

도면16


1600
1602
클라이언트(들)
통신 프레임워크
1604
서버(들)
1606
1608
클라이언트 데이터 저장소(들)
1610
서버 데이터 저장소(들)