SAT/SMT tutorial
SMT 해결기 내부

이우석
서울대학교 프로그래밍 연구실

$$i = j + 3 \wedge f(i + 3) \neq f(j + 6)$$

$$i = j + 3 \wedge f(i+3) \neq f(j+6)$$

unsat

$$x = y - 4 \wedge f(x+2) \neq f(y-1)$$

$$x = y - 4 \wedge f(x+2) \neq f(y-1)$$

sat if

$$x = -2$$

$$y = 2$$

$$f(0) = 1$$

$$f(1) = 3$$

발표의 목표

SMT 식의 만족해를 찾는
핵심 알고리즘 소개

Satisfiability Modulo Theory(SMT)

- 만족 할당 찾기(satisfiability)는 주어진 식을 참으로 만드는 할당을 찾는 문제

- 논리식이 명제논리식(propositional logic)라면, (SAT)
  - 각 변수에 참, 혹은 거짓을 값으로 할당

- 논리식이 1차논리(first-order logic)라면, (SMT)
  - 각 변수에 특정 종류의 값들을 할당(실수, 정수, 등등)

SMT는 어떻게 푸나?

- 기본적으로 SAT 해결기에 의존
- + 표현하는 식에 관련된 정보가 관여됨

Theory

Theory?

- 구성요소(Signature) + 공리(Axiom)

- 모르는 함수가 낀 방정식
  (Equality with Uninterpreted function)
  - 구성요소 : $\{f, g, h, \cdots, =\}$
  - 공리 $\forall x.x = x$

$$\forall x.y.x = y \rightarrow y = x$$

$$\forall x.y.z.x = y \wedge y = z \rightarrow x = z$$

$$\forall \bar{x}.\bar{y}.(\bigwedge_{i=1}^{n} x_i = y_i) \rightarrow f(\bar{x}) = f(\bar{y})$$

- 선형 산술 (linear arithmetic)
  - 구성요소 : $\{0, 1, +, -, \times, \geq, =\}$
  - 공리
    - + 결합법칙
    - + 항등원
    - + 역원
    - + 교환법칙

- 배열
  - 구성요소 : $\{read(..), write(..), =\}$
  - 공리

$$\forall a \forall i \forall v(read(write(a,i,v),i) = v)$$

$$\forall a \forall i \forall j \forall v(i \neq j \rightarrow read(write(a,i,v), j) = read(a, j))$$

- 그 외
  - Bitvector
  - Recursive datatypes
  - …

SMT식 풀기 방법 1 : eager approach

- 문제를 SAT 문제로 바꿔서 푼다
- 왜 eager?
  - 모든 theory 정보를 처음부터 이용하니까
- 특성
  - 잘만들어진 SAT solver를 바로 사용할 수 있음
  - 각 theory마다 각기 다른 식 변환 방식 필요

Eager approach : EUF를 예로

- 먼저 함수를 없앤다
  - f(a), f(b), f(c) … 를 A,B,C로 바꿈
  - 다음 식을 덧붙임

$$a = b \rightarrow A = B$$

$$a = c \rightarrow A = C$$

$$b = c \rightarrow B = C$$

- 각 변수들을 비트열로 표현하여 부울식으로 바꿈

Eager approach 예제

$$\underbrace{g(a)=c}_{1}\wedge\underbrace{g(b)=d}_{2}\wedge\underbrace{a=b}_{3}\wedge\underbrace{c\neq d}_{4}$$

함수 제거

$$\underbrace{A=c}_{1}\wedge\underbrace{B=d}_{2}\wedge\underbrace{a=b}_{3}\wedge\underbrace{c\neq d}_{4}\wedge\underbrace{(a=b\rightarrow A=B)}_{5}$$

CNF로

$$\underbrace{A=c}_{1}\wedge\underbrace{B=d}_{2}\wedge\underbrace{a=b}_{3}\wedge\underbrace{c\neq d}_{4}\wedge\underbrace{(a\neq b\vee A=B)}_{5}$$

부울식으로

$$\underbrace{((A_0\wedge c_0)\vee(\overline{A_0}\wedge\overline{c_0}))\wedge((A_1\wedge c_1)\vee(\overline{A_1}\wedge\overline{c_1}))\wedge((A_2\wedge c_2)\vee(\overline{A_2}\wedge\overline{c_2}))}_{1}\ldots$$

만족 불가능!

Lazy approach

$$\underbrace{g(a)=c}_{1}\wedge\underbrace{(f(g(a))\neq f(c)}_{2}\underbrace{\vee g(a)=d)}_{3}\wedge\underbrace{c\neq d}_{4}$$

- SAT solver에게 $1\wedge(2\vee 3)\wedge 4$ 를 해결하라고 던져줌
- SAT solver가 {1,2,4} 가 참이면 된다고 알려줌.
- 그러나 1과 2가 동시에 참이 될 수 없다고 Theory solver가 알려줌

Lazy approach

$$\underbrace{g(a)=c}_{1}\wedge\underbrace{(f(g(a))\neq f(c)}_{2}\underbrace{\vee g(a)=d)}_{3}\wedge\underbrace{c\neq d}_{4}$$

- SAT solver에게 $1\wedge(2\vee 3)\wedge 4\wedge(\bar{1}\vee\bar{2}\vee\bar{4})$

를 해결하라고 던져줌

- SAT solver가 $\{1,\bar{2},3,4\}$가 참이면 된다고 알려줌.
- 그러나 1,3,4가 동시에 참이 될 수 없다고 Theory solver가 알려줌

Lazy approach

$$\underbrace{g(a)=c}_{1}\wedge\underbrace{(f(g(a))\neq f(c)}_{2}\underbrace{\vee g(a)=d)}_{3}\wedge\underbrace{c\neq d}_{4}$$

- SAT solver에게

  $1\wedge(2\vee 3)\wedge 4\wedge(\bar{1}\vee\bar{2}\vee\bar{4})\wedge(\bar{1}\vee 2\vee\bar{3}\vee\bar{4})$

  를 해결하라고 던져줌

- SAT solver가 이 식이 만족될 수 없다고 함

Lazy approach

- 특성
  - Modular 하고 유연함
  - Theory 정보가 해 탐색을 안내하지 않음
- 실제로는 많은 최적화가 일어남

여러 theory 결합

- 소프트웨어 검증은 산술, 배열, 비트벡터 등에 관한 표현식들을 필요로 한다.
- 예)

$$a = b + 2 \wedge B[a+1] \leftarrow 4 \wedge (A[b+3] = 2 \vee f(a-1) \neq f(b+1))$$

- 각 theory의 decision procedure를 결합하여 해결

여러 theory 결합

$$
\begin{aligned}
f(f(x)-f(y)) &= a \\
f(0) &= a+2 \\
x &= y
\end{aligned}
$$

- 두 Theory가 관계되어 있음 :
  - 실수 산술 연산 $(Arithmetic)$
  - 모르는 함수가 낀 방정식(EUF)

- 1단계 : 변수가 아닌 literal들에게 새로운 이름을 주어 하나의 theory에만 관계되게 한다.

$$f(f(x)-f(y))=a \Rightarrow \begin{aligned} f(e_1) &= a \\ e_1 &= f(x)-f(y) \end{aligned} \Rightarrow \begin{aligned} f(e_1) &= a \\ e_1 &= e_2-e_3 \\ e_2 &= f(x) \\ e_3 &= f(y) \end{aligned}$$

EUF 부분 만들기

- 그렇게 하는 이유?
  - 변수와 '=' 는 모든 theory 공통 구성요소

- 1단계 : 변수가 아닌 literal들을 새로운 변수로 대치하여 하나의 theory에만 관계되게 한다.

$$f(0) = a + 2 \quad \Rightarrow \quad \begin{aligned} f(e_4) &= a + 2 \\ e_4 &= 0 \end{aligned} \quad \Rightarrow \quad \begin{aligned} f(e_4) &= e_5 \\ e_4 &= 0 \\ e_5 &= a + 2 \end{aligned}$$

Arithmetic 부분 만들기

- 2단계 : 각 theory solver가 식 만족여부를 검사하고 공통된 변수($e_1, e_2, e_3, e_4, e_5, a$)의 방정식을 서로 교환

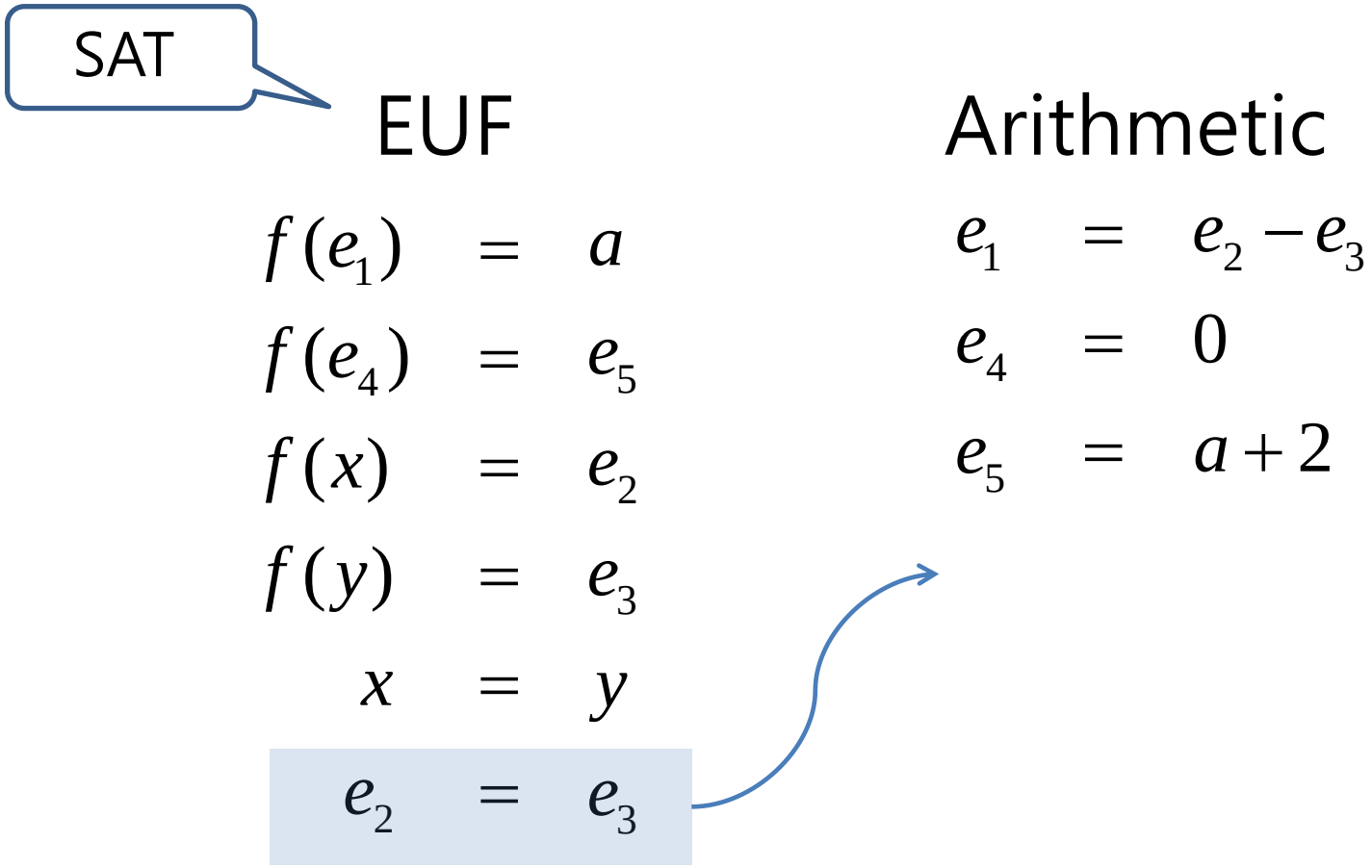
EUF

$$
\begin{aligned}
f(e_1) &= a \\
f(e_4) &= e_5 \\
f(x) &= e_2 \\
f(y) &= e_3 \\
x &= y
\end{aligned}
$$

Arithmetic

$$
\begin{aligned}
e_1 &= e_2 - e_3 \\
e_4 &= 0 \\
e_5 &= a + 2
\end{aligned}
$$

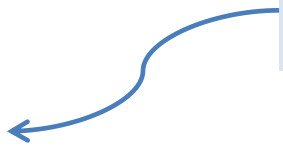
- 2단계 : 각 theory solver가 식 만족여부를 검사하고 공통된 변수($e_1, e_2, e_3, e_4, e_5, a$)의 방정식을 서로 교환



- 2단계 : 각 theory solver가 식 만족여부를 검사하고 공통된 변수($e_1, e_2, e_3, e_4, e_5, a$)의 방정식을 서로 교환

SAT

EUF			Arithmetic		
$f(e_1)$	$=$	a	e_1	$=$	$e_2 - e_3$
$f(e_4)$	$=$	e_5	e_4	$=$	0
$f(x)$	$=$	e_2	e_5	$=$	$a+2$
$f(y)$	$=$	e_3	e_2	$=$	e_3
x	$=$	y	e_1	$=$	e_4



- 2단계 : 각 theory solver가 식 만족여부를 검사하고 공통된 변수($e_1, e_2, e_3, e_4, e_5, a$)의 방정식을 서로 교환

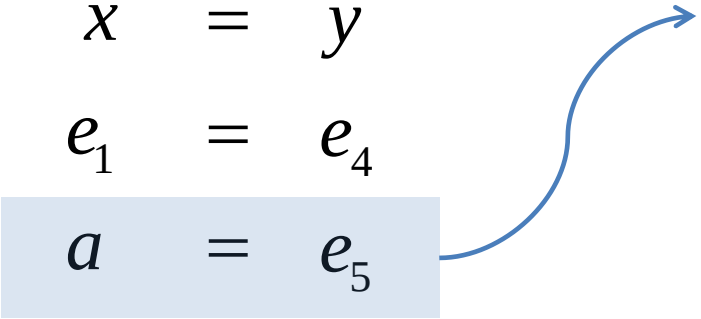
SAT

EUF

$$
\begin{aligned}
f(e_1) &= a \\
f(e_4) &= e_5 \\
f(x) &= e_2 \\
f(y) &= e_3 \\
x &= y \\
e_1 &= e_4 \\
a &= e_5
\end{aligned}
$$

Arithmetic

$$
\begin{aligned}
e_1 &= e_2 - e_3 \\
e_4 &= 0 \\
e_5 &= a + 2 \\
e_2 &= e_3
\end{aligned}
$$



- 2단계 : 각 theory solver가 식 만족여부를 검사하고 공통된 변수($e_1, e_2, e_3, e_4, e_5, a$)의 방정식을 서로 교환

UNSAT

EUF			Arithmetic		
$f(e_1)$	$=$	a	e_1	$=$	$e_2 - e_3$
$f(e_4)$	$=$	e_5	e_4	$=$	0
$f(x)$	$=$	e_2	e_5	$=$	$a+2$
$f(y)$	$=$	e_3	e_2	$=$	e_3
x	$=$	y	a	$=$	e_5
e_1	$=$	e_4			

결론

- SMT 는 SAT 해결기를 이용하여 풀 수 있다
  - 두 가지 방식 : Eager와 Lazy
  - 여러 Theory가 관여된 경우 : Theory solver들끼리 서로 정보 교환하면서 문제를 푼다.

- 더 알고 싶으신 분은 참고자료를 보세요.

참고자료

- Albert Oliveras, SMT Theory and DPLL(T), 1st SAT/SMT solver summer school 2011
- Aaron Bradley, Zohar Manna, The Calculus of Computation
- Armin Biere et al., Handbook of satisfiability