Verificare Formala

Prezentare Generala

Mădălina Erașcu

West University of Timișoara
Faculty of Mathematics and Informatics
Department of Computer Science

Bazat pe suportul de curs: *Satisfiability Checking (Erika Ábrahám), RTWH Aachen*
Bazat pe cartile: (1) The calculus of computation - Z. Manna, A. Bradley; (2) Decision Procedures - An algorithmic Point of View - D. Kroening, O. Strichman

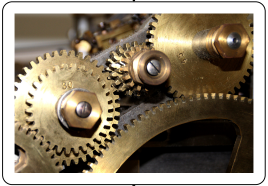
WS 2023/2024

Organizare

Vezi Fisa Disciplinei

Despre ce este acest curs?

Formula fara cuantificatori logici
(Quantifier-free logical formula)

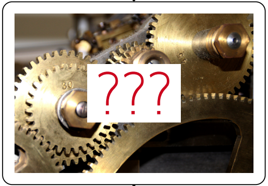


Rezolvitor (Solver)

Satisfiability of the input formula
(Satisfiabilitatea formulei de intrare)

Despre ce este acest curs?

Formula fara cuantificatori logici
(Quantifier-free logical formula)



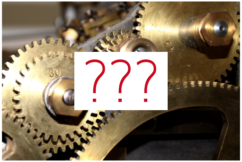
Rezolvitor (Solver)

Satisfiability of the input formula
(Satisfiabilitatea formulei de intrare)

Despre ce este acest curs?

???

Formula fara cuantificatori logici
(Quantifier-free logical formula)



Rezolvitor (Solver)

Satisfiability of the input formula
(Satisfiabilitatea formulei de intrare)

???

Problema de satisfactie booleana ...

Problema de satisfiabilitate (SAT) pentru logica propozitiilor

Dandu-se o formula care combina propozitii atomice folosind operatorii booleni "si" ($\wedge$), "sau" ($\vee$) si "nu" ($\neg$), decideti daca exista valori de adevar pentru atomii din formula astfel incat formula se evalueaza la adevarat.

Example

Formula: $(a \vee \neg b) \wedge (\neg a \vee b \vee c)$

Atribuire pt. satisfiabilitate: $a = true, \quad b = false, \quad c = true$

(satisfying assignment):

Este probabil cea mai cunoscuta problema NP-completa [Cook, 1971] [Levin, 1973].

...si extinderea sa la teorii

Problema de satisfiabilitate in teorii logice (satisfiability modulo theory) (definitie informala)

Dandu-se o combinatie booleana de constrangeri din anumite teorii, decideti daca putem inlocui valorile pentru variabilele din teorie astfel incat formula sa fie evaluata ca fiind adevarata (SAT).

Un exemplu aritmetic real neliniar

Formula:	$\underset{x}{\exists} \left(x^2 + 1 \geq 0 \wedge x < 0\right)$
Satisfying assignment:	$x = -1$

Probleme grele ... aritmetica intreaga neliniara este chiar indecidabila.

Ce este logica formala?

- O logica (formala)

Ce este logica formala?

- O logica (formala) defineste un cadru pentru inferenta si rationament corect.

Ce este logica formala?

- O logica (formala) defineste un cadru pentru inferenta si rationament corect.
- studiata in, de exemplu,

Ce este logica formala?

- O logica (formala) defineste un cadru pentru inferenta si rationament corect.
- studiata in, de exemplu, filosofie, matematica, informatica.

Ce este logica formala?

- O logica (formala) defineste un cadru pentru inferenta si rationament corect.
- studiata in, de exemplu, filosofie, matematica, informatica.
- Un sistem logic defineste

Ce este logica formala?

- O logica (formala) defineste un cadru pentru inferenta si rationament corect.
- studiata in, de exemplu, filosofie, matematica, informatica.
- Un sistem logic defineste
  - forma formulelor logice (sintaxa) si

Ce este logica formala?

- O logica (formala) defineste un cadru pentru inferenta si rationament corect.
- studiata in, de exemplu, filosofie, matematica, informatica.
- Un sistem logic defineste
  - forma formulelor logice (sintaxa) si
  - un set de axiome si un set de reguli de inferenta.

Ce este logica formala?

- O logica (formala) defineste un cadru pentru inferenta si rationament corect.
- studiata in, de exemplu, filosofie, matematica, informatica.
- Un sistem logic defineste
  - forma formulelor logice (sintaxa) si
  - un set de axiome si un set de reguli de inferenta.
- Care este valoarea de adevar a unei formule logice?

Ce este logica formala?

- O logica (formala) defineste un cadru pentru inferenta si rationament corect.
- studiata in, de exemplu, filosofie, matematica, informatica.
- Un sistem logic defineste
  - forma formulelor logice (sintaxa) si
  - un set de axiome si un set de reguli de inferenta.
- Care este valoarea de adevar a unei formule logice?
  - A structure for a logical system gives meaning (semantics) to the formulas.
    O structura pentru un sistem logic da semnificatie (semantica) formulelor.
  - Sistemul logic permite a se deriva sensul formulelor.

Ce este logica formala?

- O logica (formala) defineste un cadru pentru inferenta si rationament corect.
- studiata in, de exemplu, filosofie, matematica, informatica.
- Un sistem logic defineste
  - forma formulelor logice (sintaxa) si
  - un set de axiome si un set de reguli de inferenta.
- Care este valoarea de adevar a unei formule logice?
  - A structure for a logical system gives meaning (semantics) to the formulas.
    O structura pentru un sistem logic da semnificatie (semantica) formulelor.
  - Sistemul logic permite a se deriva sensul formulelor.
- Proprietati importante ale sistemelor logice:

Ce este logica formala?

- O logica (formala) defineste un cadru pentru inferenta si rationament corect.
- studiata in, de exemplu, filosofie, matematica, informatica.
- Un sistem logic defineste
  - forma formulelor logice (sintaxa) si
  - un set de axiome si un set de reguli de inferenta.
- Care este valoarea de adevar a unei formule logice?
  - A structure for a logical system gives meaning (semantics) to the formulas.
    O structura pentru un sistem logic da semnificatie (semantica) formulelor.
  - Sistemul logic permite a se deriva sensul formulelor.
- Proprietati importante ale sistemelor logice:
  - consistenta
  - corectitudine (soundness)
  - completitudine (completeness)

Viziune istorica asupra logicii

Dezvoltarea istorica porneste de la:

logica informala (argumente de limbaj natural) *pana la*

logica formala (argumente de limbaj formal)

Viziune istorica asupra logicii

Dezvoltarea istorica porneste de la:

logica informala (argumente de limbaj natural) *pana la*

logica formala (argumente de limbaj formal)

- logica filosofica
  - de la 500 BC pana in secolul al-19-lea
- logica simbolica
  - mijlocul pana spre sfarsitul secolului al-19-lea
- logica matematica
  - sfarsitul secolului al-19-lea pana la mijlocul secolului al-20-lea
- logica in informatica

Logica are un impact profund asupra informaticii. Cateva example:

Logica in informatica

Logica are un impact profund asupra informaticii. Cateva example:

- logica propozitionala – fundamentul calculatoarelor si circuitelor
- baze de date – limbaje de interogare (query languages)
- limbaje de prgramare (de exemplu, Prolog)
- specificare si verificare
- ...

Logica in informatica

- logica propositionala
- logica predicatelor de ordinul 1 (first order logic)
- logica predicatelor de ordin superior (higher order logic)
- logica temporala (temporal logic)
- ...

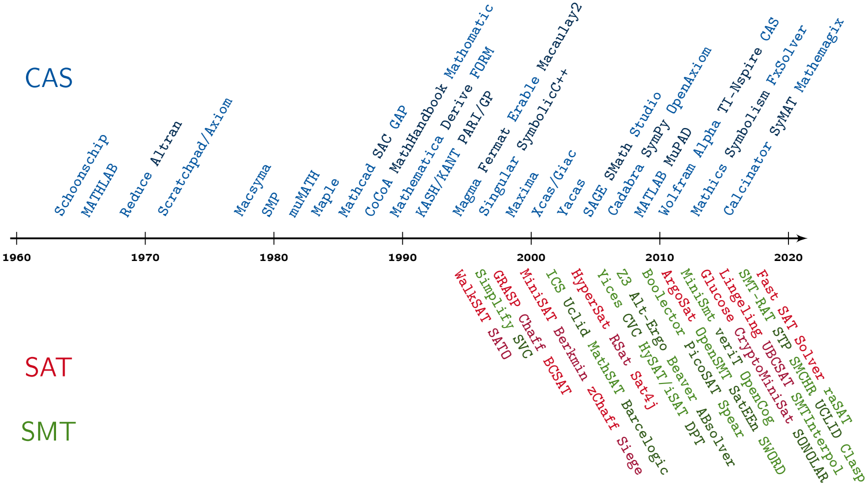
Verificarea satisfiabilitatii: Cateva repere

Decision procedures for first-order logic over arithmetic theories
in mathematical logic

1940 Computer architecture development

Year	CAS (Symbolic Computation)	SAT (propositional logic)	SMT (SAT modulo theories)
		Enumeration	
1960	Computer algebra systems (CAS)	DP (resolution) DPLL (propagation)	
1970	Gröbner bases CAD (cylindrical algebraic decomposition)	NP-completeness	Decision procedures for combined theories
1980	FGLM algorithm Partial CAD Comprehensive Gröbner bases	Conflict-directed backjumping	
2000	Virtual substitution	CDCL Watched literals Clause learning/forgetting Variable ordering heuristics Restarts	DPLL(T) Equalities Uninterpreted functions Bit-vectors Array theory Arithmetic theories
2010			
2015	Truth table invariant CAD		

Verificarea satisfiabilitatii: Dezvoltarea instrumentelor (lista nu este exhaustiva)



Verificarea satisfiabilitatii pentru logica propozitionala

Poveste de succes: Rezolvarea satisfiabilitatii problemelor SAT-solving

- Problemele practice cu milioane de variabile sunt rezolvabile.
- Utilizat frecvent in diferite domenii de cercetare pentru, de exemplu, analiza, sinteza si optimizare.
- De asemenea, este utilizat in mod frecvent in industrie pentru, de exemplu, proiectarea si verificarea circuitelor digitale.

Verificarea satisfiabilitatii pentru logica propozitionala

Poveste de succes: Rezolvarea satisfiabilitatii problemelor SAT-solving

- Problemele practice cu milioane de variabile sunt rezolvabile.
- Utilizat frecvent in diferite domenii de cercetare pentru, de exemplu, analiza, sinteza si optimizare.
- De asemenea, este utilizat in mod frecvent in industrie pentru, de exemplu, proiectarea si verificarea circuitelor digitale.

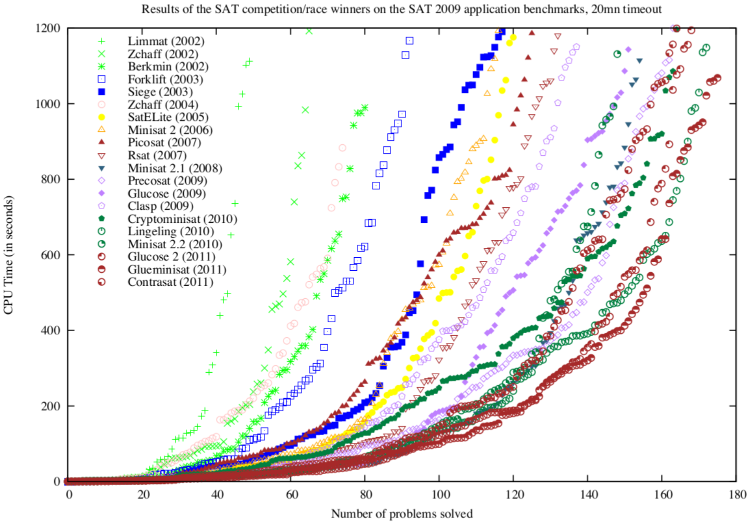
Suport din comunitate:

- Limbaj de intrare standardizat, multe benchmarkuri disponibile.
- Competitii din 2002.

  Competitia SAT 2016: 6 curse (track), 29 solvere in cursa principala.

  Forumul SAT Live! este o platforma a comunitatii dedicata conferintelor, jurnalelor, etc.

Impresia asupra dezvoltatii domeniului SAT



Sursa: Jarvisalo, Le Berre, Roussel, Simon. *The International SAT Solver Competitions*. AI Magazine, 2012.

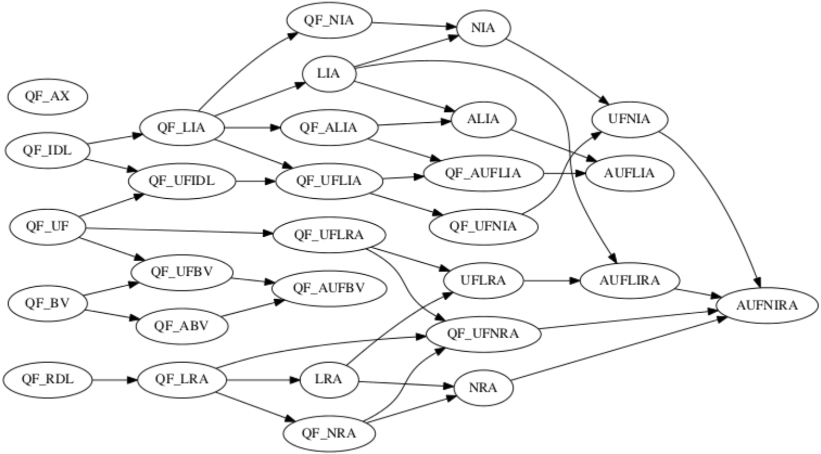
Rezolvarea satisfiabilitatii in teorii (Satisfiability modulo theories solving)

- Logica propozitionala este uneori prea inexpresiva pentru modelare.
- Avem nevoie de o logica mai expresiva si de proceduri de decizie (decision procedures, algorithms) pentru acestea.
- Tipuri de logica:
  fragmente de logica predicatelor de ordinul 1 fara cuantificatori logici (quantifier-free fragments of first-order logic over various theories).
- Focusul nostru: SAT-modulo-theories (SMT) solving.

Rezolvarea satisfiabilitatii in teorii (Satisfiability modulo theories solving)

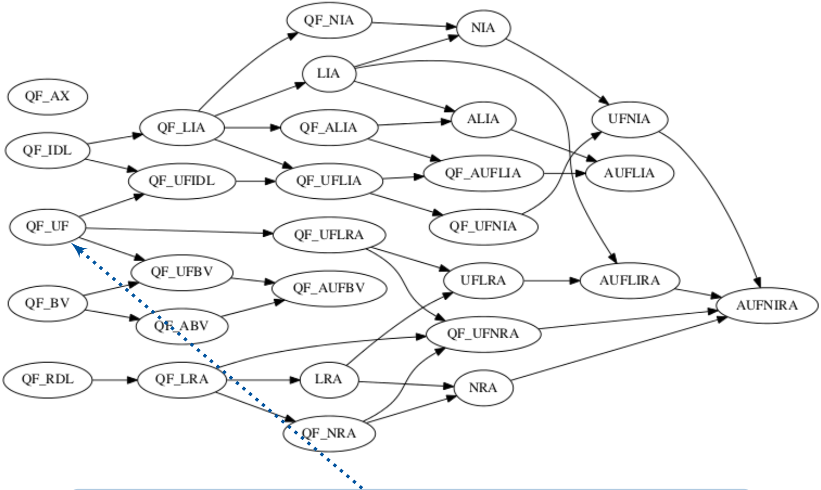
- Logica propozitionala este uneori prea inexpresiva pentru modelare.
- Avem nevoie de o logica mai expresiva si de proceduri de decizie (decision procedures, algorithms) pentru acestea.
- Tipuri de logica:
  fragmente de logica predicatelor de ordinul 1 fara cuantificatori logici (quantifier-free fragments of first-order logic over various theories).
- Focusul nostru: SAT-modulo-theories (SMT) solving.
- SMT-LIB ca limbaj de intrare standard since 2004.
- Competitii din 2005.
- competitia SMT-COMP 2016:
  - 4 curse (tracks), 41 teorii logice.
  - logica reala liniara fara cuantificatori logici (QF linear real arithmetic): $7 + 2$ solvere, 1626 benchmarks.
  - logica intreaga liniara fara cuantificatori logici (QF linear integer arithmetic): $6 + 2$ solvere, 5839 benchmarks.
  - logica reala neliniara fara cuantificatori logici (QF non-linear real arithmetic): $5 + 1$ solvere, 10245 benchmarks.
  - logica intreaga neliniara fara cuantificatori logici (QF non-linear integer arithmetic): $7 + 1$ solvere, 8593 benchmarks.

Teorii care se regasesc in SMT-LIB



Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

Teorii care se regasesc in SMT-LIB

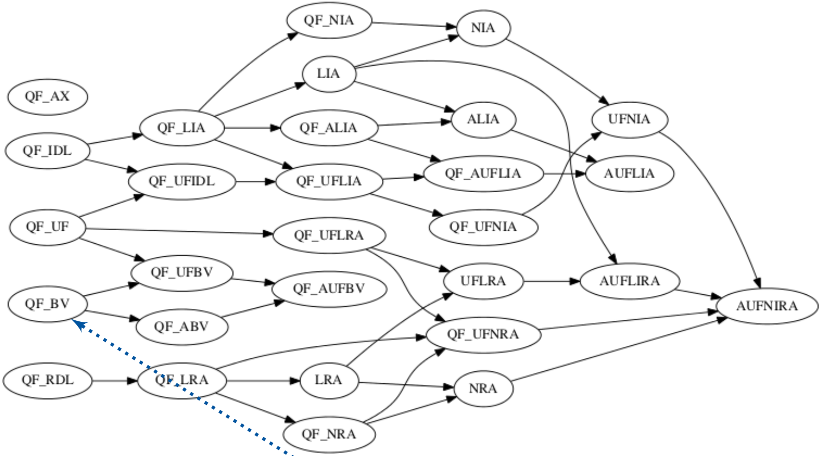


Quantifier-free equality logic with uninterpreted functions

$$(a = c \wedge b = d) \rightarrow f(a, b) = f(c, d)$$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

Teorii care se regasesc in SMT-LIB

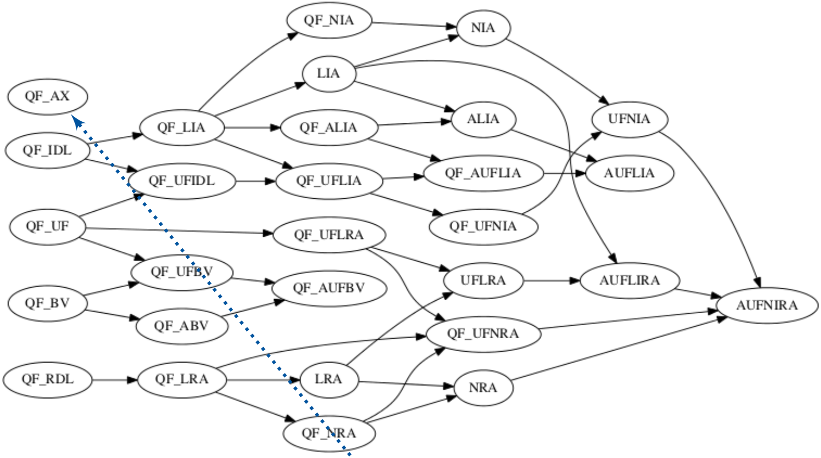


Quantifier-free bit-vector arithmetic

$a + b \geq 0 \wedge (\, a|b\,) \leq (\, a\&b\,)$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

Teorii care se regasesc in SMT-LIB

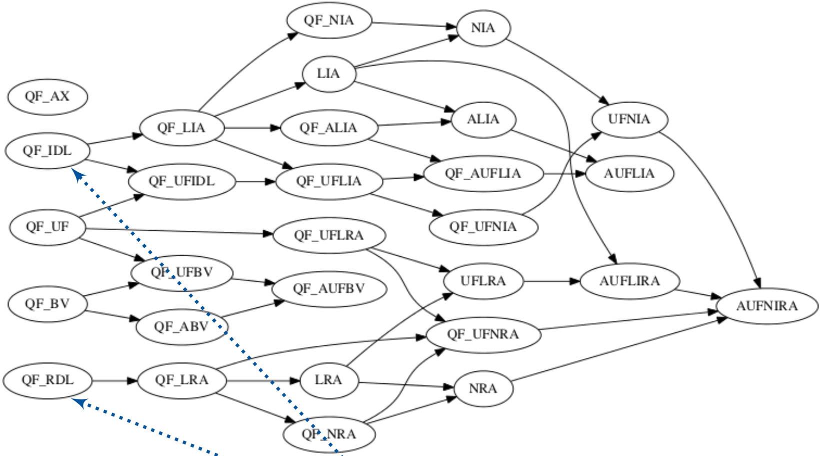


Quantifier-free array theory

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

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

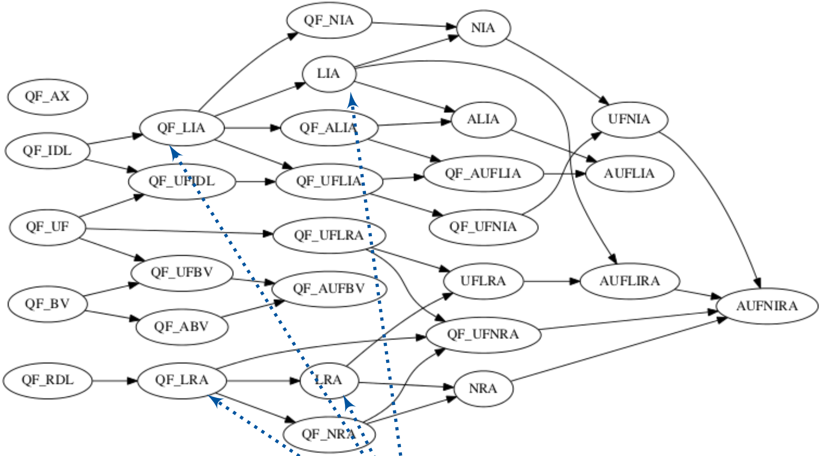
Teorii care se regasesc in SMT-LIB



Quantifier-free integer/rational difference logic
$x - y \geq 0 \vee x - z < 0$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

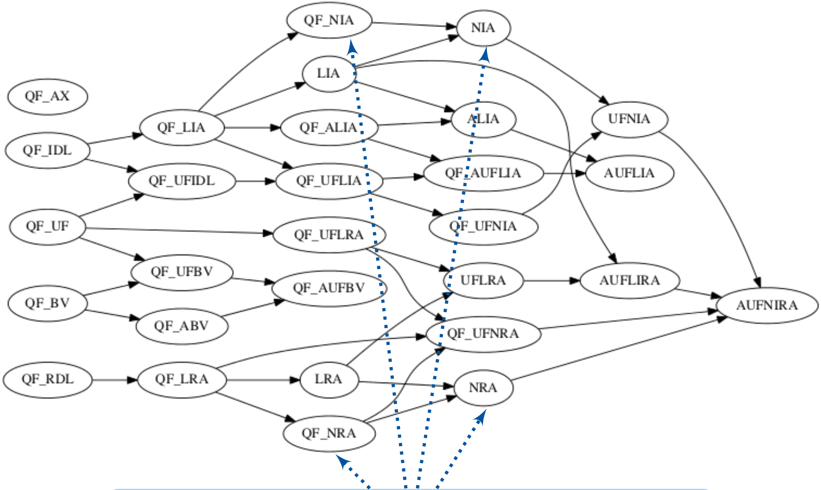
Teorii care se regasesc in SMT-LIB



(Quantifier-free) real/integer linear arithmetic
$4x + 7y = 8 \wedge (y = 0 \vee x > y)$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

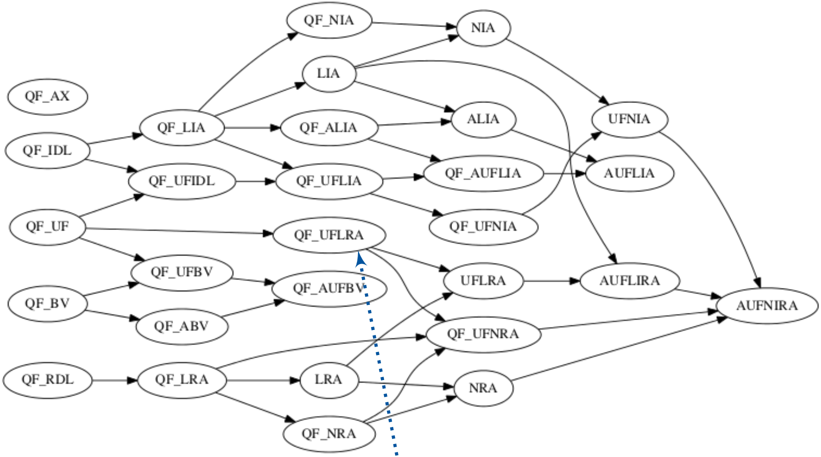
Teorii care se regasesc in SMT-LIB



(Quantifier-free) real/integer non-linear arithmetic
$x^2 + 2xy + y^2 > 0 \vee (x \geq 1 \wedge xz + yz^2 = 0)$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

Teorii care se regasesc in SMT-LIB

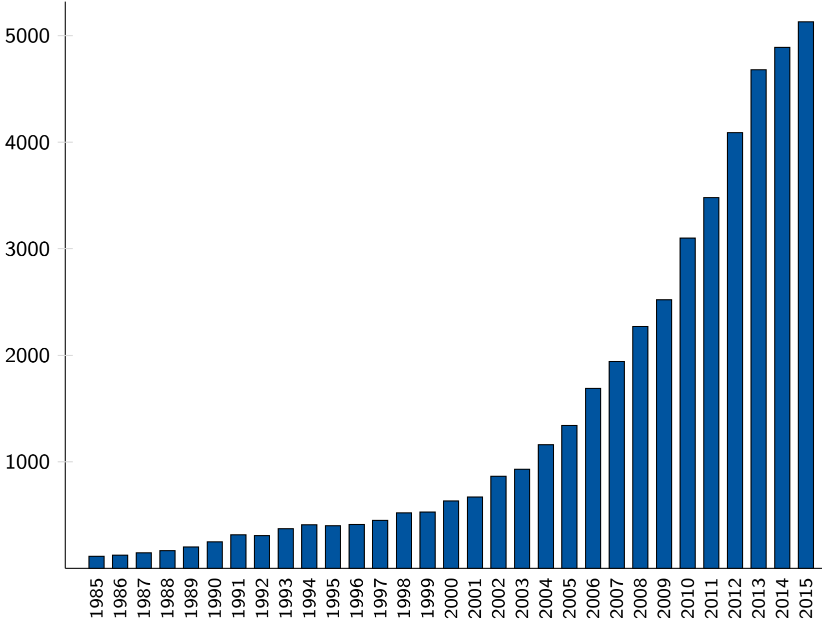


Combined theories

$2f(x) + 5y > 0 \land \neg(f(x) = y \lor x + 2y = 0)$

Source: `http://smtlib.cs.uiowa.edu/logics.shtml`

Cautare Google Scholar pentru “SAT modulo theories”



Cautare Google Scholar pentru "SAT modulo theories"

In 2021: 29000 citari!

Prezentarea generala a teoriilor logice din acest curs

Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor **(propositional logic)**

Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor **(propositional logic)** $(x \vee y) \wedge (\neg x \vee y)$

Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor (propositional logic) $\quad (x \vee y) \wedge (\neg x \vee y)$

Logica vectorilor de biti (Bitvector arithmetic)

Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor (propositional logic) $(x \vee y) \wedge (\neg x \vee y)$

Logica vectorilor de biti (Bitvector arithmetic) $a \,\&\, (b{<<}1)$

Egalitate (Equality)

Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor (propositional logic)	$(x \vee y) \wedge (\neg x \vee y)$
Logica vectorilor de biti (Bitvector arithmetic)	$a \,\&\, (b << 1)$
Egalitate (Equality)	$(x = y \wedge y \neq z) \rightarrow (x \neq z)$

Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor (propositional logic)	$(x \vee y) \wedge (\neg x \vee y)$
Logica vectorilor de biti (Bitvector arithmetic)	$a \ \& \ (b<<1)$
Egalitate (Equality)	$(x = y \wedge y \neq z) \rightarrow (x \neq z)$
Functii neinterpretate	

Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor **(propositional logic)**	$(x \vee y) \wedge (\neg x \vee y)$
Logica vectorilor de biti **(Bitvector arithmetic)**	$a \,\&\, (b{<<}1)$
Egalitate **(Equality)**	$(x = y \wedge y \neq z) \rightarrow (x \neq z)$
Functii neinterpretate **(Uninterpreted functions)**	$(F(x){=}F(y) \wedge y{=}z) \rightarrow F(x){=}F(z)$

Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor **(propositional logic)**	$(x \vee y) \wedge (\neg x \vee y)$
Logica vectorilor de biti **(Bitvector arithmetic)**	$a\ \&\ (b{<}{<}1)$
Egalitate **(Equality)**	$(x = y \wedge y \neq z) \rightarrow (x \neq z)$
Functii neinterpretate **(Uninterpreted functions)**	$(F(x){=}F(y) \wedge y{=}z) \rightarrow F(x){=}F(z)$
Aritmetica intreaga/reala liniara	

Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor (propositional logic)	$(x \vee y) \wedge (\neg x \vee y)$
Logica vectorilor de biti (Bitvector arithmetic)	$a \,\&\, (b<<1)$
Egalitate (Equality)	$(x = y \wedge y \neq z) \to (x \neq z)$
Functii neinterpretate (Uninterpreted functions)	$(F(x)=F(y) \wedge y=z) \to F(x)=F(z)$
Aritmetica intreaga/reala liniara (Linear real/integer arithmetic)	$2x + y > 0 \wedge x + y \leq 0$

Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor (propositional logic)	$(x \vee y) \wedge (\neg x \vee y)$
Logica vectorilor de biti (Bitvector arithmetic)	$a \mathbin{\&} (b<<1)$
Egalitate (Equality)	$(x = y \wedge y \neq z) \rightarrow (x \neq z)$
Functii neinterpretate (Uninterpreted functions)	$(F(x){=}F(y) \wedge y{=}z) \rightarrow F(x){=}F(z)$
Aritmetica intreaga/reala liniara (Linear real/integer arithmetic)	$2x + y > 0 \wedge x + y \leq 0$ $2x = 1$

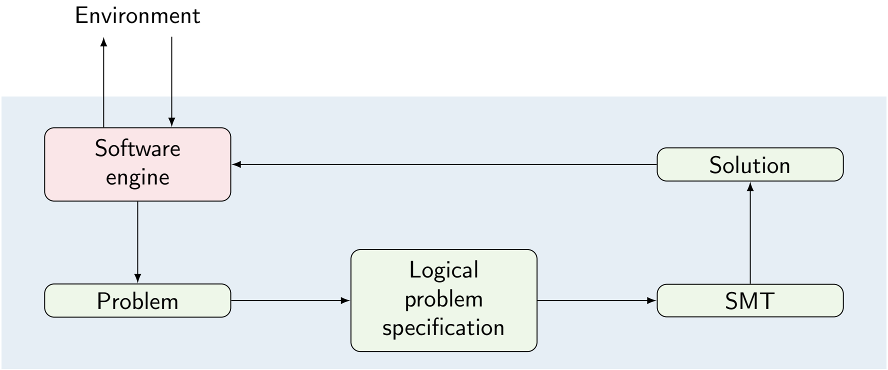
Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor (propositional logic)	$(x \vee y) \wedge (\neg x \vee y)$
Logica vectorilor de biti (Bitvector arithmetic)	$a \,\&\, (b{<<}1)$
Egalitate (Equality)	$(x = y \wedge y \neq z) \rightarrow (x \neq z)$
Functii neinterpretate (Uninterpreted functions)	$(F(x){=}F(y) \wedge y{=}z) \rightarrow F(x){=}F(z)$
Aritmetica intreaga/reala liniara (Linear real/integer arithmetic)	$2x + y > 0 \wedge x + y \leq 0$ $2x = 1$
Aritmetica reala neliniara ((Non-linear) real arithmetic)	

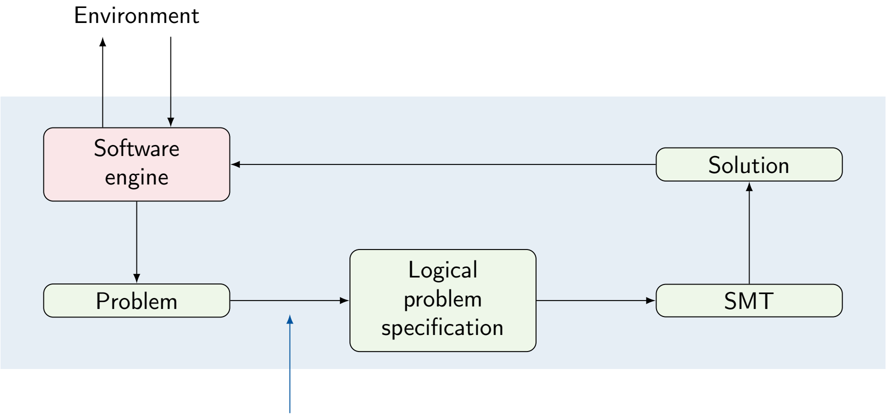
Prezentarea generala a teoriilor logice din acest curs

Logica propozitiilor (propositional logic)	$(x \vee y) \wedge (\neg x \vee y)$
Logica vectorilor de biti (Bitvector arithmetic)	$a\ \&\ (b{<<}1)$
Egalitate (Equality)	$(x = y \wedge y \neq z) \rightarrow (x \neq z)$
Functii neinterpretate (Uninterpreted functions)	$(F(x){=}F(y) \wedge y{=}z) \rightarrow F(x){=}F(z)$
Aritmetica intreaga/reala liniara (Linear real/integer arithmetic)	$2x + y > 0 \wedge x + y \leq 0$ $2x = 1$
Aritmetica reala neliniara ((Non-linear) real arithmetic) ((Non-linear) real arithmetic)	$x^2 + 2xy + y^2 < 0$

Arhitectura unui sistem care incorporeaza solvere SAT/SMT



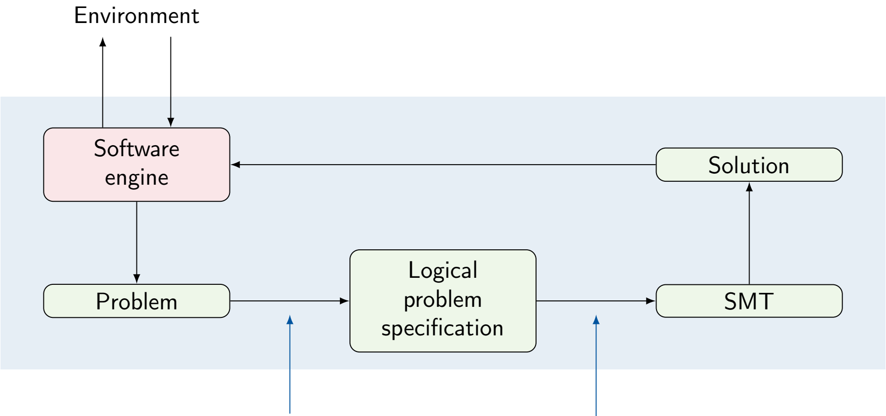
Arhitectura unui sistem care incorporeaza solvere SAT/SMT



Formalizarea (Encoding): standardul SAT/SMT-LIB

formalizarea este extrem de importanta!

Arhitectura unui sistem care incorporeaza solvere SAT/SMT



Formalizarea (Encoding): standardul SAT/SMT-LIB
formalizarea este extrem de importanta!

standard input language → free solver choice

Exemple de aplicatii: verificarea hardware

Problema 1: Dandu-se 2 circuite, sunt acestea echivalente?

Problema 2: Dandu-se un circuit si o proprietate, verifica circuitul proprietatea?

Problema 3: Dandu-se un circuit partial specificat de o componenta de tip black-box (in stadiul de design timpuriu) si o specificatie, este circuitul partial realizabil, adica exista o implementare a componentei de tip black-box astfel incat circuitul satisface proprietatea?

Multi producatori de hardware dezvolta si utilizeaza solvere SAT proprii pentru aceste sarcini.

Example de aplicatuu: executia simbolica

Program 1.2.1 A recursion-free program with bounded loops and an SSA unfolding.

```
int Main(int x, int y)
{
  if (x < y)
     x = x + y;
  for (int i = 0; i < 3; ++i) {
       y = x + Next(y);
  }
  return x + y;
}

int Next(int x) {
  return x + 1;
}
```

```
int Main(int x0, int y0)
{
  int x1;
  if (x0 < y0)
     x1 = x0 + y0;
  else
     x1 = x0;
  int y1 = x1 + y0 + 1;
  int y2 = x1 + y1 + 1;
  int y3 = x1 + y2 + 1;
  return x1 + y3;
}
```

$$\exists x_1, y_1, y_2, y_3 \begin{pmatrix} (x_0 < y_0 \implies x_1 = x_0 + y_0) \wedge (\neg(x_0 < y_0) \implies x_1 = x_0) \wedge \\ y_1 = x_1 + y_0 + 1 \wedge y_2 = x_1 + y_1 + 1 \wedge y_3 = x_1 + y_2 + 1 \wedge \\ result = x_1 + y_3 \end{pmatrix}$$

Source: Nikolaj Bjørner and Leonardo de Moura. ***Applications of SMT solvers to Program Verification*****.**

Rough notes for SSFT 2014.

Example de aplicatii: Verificarea modelului delimitat (Bounded model checking)

Problema: Dandu-se un program (automat, circuit, sistem de rescriere, etc.), gasiti o ramura de executie a programului de lungime cel mult k care conduce la o stare care satisface o anumita proprietate (folosita pentru detectia impartirii cu 0, violarea cerintelor functiilor, etc.).

Example de aplicatii: Bounded model checking for C/C++

Carnegie Mellon

CBMC Homepage

Bounded Model Checking for Software

About CBMC

CBMC is a Bounded Model Checker for C and C++ programs. It supports C89, C99, most of C11 and most compiler extensions provided by gcc and Visual Studio. It also supports SystemC using Scoot. We have recently added experimental support for Java Bytecode.

CBMC verifies array bounds (buffer overflows), pointer safety, exceptions and user-specified assertions. Furthermore, it can check C and C++ for consistency with other languages, such as Verilog. The verification is performed by unwinding the loops in the program and passing the resulting equation to a decision procedure.

While CBMC is aimed for embedded software, it also supports dynamic memory allocation using `malloc` and new. For questions about CBMC, contact Daniel Kroening.

CBMC is available for most flavours of Linux (pre-packaged on Debian, Ubuntu and Fedora), Solaris 11, Windows and MacOS X. You should also read the CBMC license.

CBMC comes with a built-in solver for bit-vector formulas that is based on MiniSat. As an alternative, CBMC has featured support for external SMT solvers since version 3.3. The solvers we recommend are (in no particular order) Boolector, MathSAT, Yices 2 and Z3. Note that these solvers need to be installed separately and have different licensing conditions.

Source: D. Kroening. **CBMC home page.** `http://www.cprover.org/cbmc/`

Example de aplicatii: Bounded model checking for C/C++

Carnegie Mellon

CBMC Homepage

Bounded Model Checking for Software

Logical encoding of finite unsafe paths

About CBMC

CBMC is a Bounded Model Checker for C and C++ programs. It supports C89, C99, most of C11 and most compiler extensions provided by gcc and Visual Studio. It also supports SystemC using Scoot. We have recently added experimental support for Java Bytecode.

CBMC verifies array bounds (buffer overflows), pointer safety, exceptions and user-specified assertions. Furthermore, it can check C and C++ for consistency with other languages, such as Verilog. The verification is performed by unwinding the loops in the program and passing the resulting equation to a decision procedure.

While CBMC is aimed for embedded software, it also supports dynamic memory allocation using `malloc` and new. For questions about CBMC, contact Daniel Kroening.

CBMC is available for most flavours of Linux (pre-packaged on Debian, Ubuntu and Fedora), Solaris 11, Windows and MacOS X. You should also read the CBMC license.

CBMC comes with a built-in solver for bit-vector formulas that is based on MiniSat. As an alternative, CBMC has featured support for external SMT solvers since version 3.3. The solvers we recommend are (in no particular order) Boolector, MathSAT, Yices 2 and Z3. Note that these solvers need to be installed separately and have different licensing conditions.

Source: D. Kroening. **CBMC home page.** `http://www.cprover.org/cbmc/`

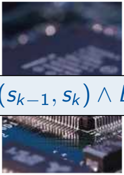
Carnegie Mellon

CBMC Homepage

Bounded Model Checking for Software

Logical encoding of finite unsafe paths

CBMC About CBMC



CBMC is a Bounded Model Checker for C and C++ programs. It supports C89, C99, most of C11 and most compiler extensions provided by gcc and Visual Studio. It also supports SystemC using Scoot. We have recently added experimental support for Java Bytecode.

Ideea formalizarii: $Init(s_0) \wedge Trans(s_0, s_1) \wedge \ldots \wedge Trans(s_{k-1}, s_k) \wedge Bad(s_0, \ldots, s_k)$

tions and user-specified assertions. Furthermore, it can check C and C++ for consistency with other languages, such as Verilog. The verification is performed by unwinding the loops in the program and passing the resulting equation to a decision procedure.

While CBMC is aimed for embedded software, it also supports dynamic memory allocation using `malloc` and new. For questions about CBMC, contact Daniel Kroening.

CBMC is available for most flavours of Linux (pre-packaged on Debian, Ubuntu and Fedora), Solaris 11, Windows and MacOS X. You should also read the CBMC license.

CBMC comes with a built-in solver for bit-vector formulas that is based on MiniSat. As an alternative, CBMC has featured support for external SMT solvers since version 3.3. The solvers we recommend are (in no particular order) Boolector, MathSAT, Yices 2 and Z3. Note that these solvers need to be installed separately and have different licensing conditions.

Source: D. Kroening. **CBMC home page.** `http://www.cprover.org/cbmc/`

Example de aplicatii: Bounded model checking for C/C++

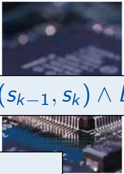
Carnegie Mellon

CBMC Homepage

Bounded Model Checking for Software

Logical encoding of finite unsafe paths

About CBMC



CBMC is a Bounded Model Checker for C and C++ programs. It supports C89, C99, most of C11 and most compiler extensions provided by gcc and Visual Studio. It also supports SystemC using Scoot. We have recently added experimental support for Java Bytecode.

Ideea formalizarii: $Init(s_0) \wedge Trans(s_0, s_1) \wedge \ldots \wedge Trans(s_{k-1}, s_k) \wedge Bad(s_0, \ldots, s_k)$

tions and user-specified assertions. Furthermore, it can check C and C++ for consistency with other languages, such as Verilog. The verification is ... passing the re...

While CBMC ... allocation using mallo...

CBMC is avail... d Fedora), Solaris 11, W...

CBMC come... Sat. As an alternative, C... 3.3. The solvers we recommend are (in no particular order) Boolector, MathSAT, Yices 2 and Z3. Note that these solvers need to be installed separately and have different licensing conditions.

Application examples:

- Error localisation and explanation
- Equivalence checking
- Test case generation
- Worst-case execution time

Source: D. Kroening. **CBMC home page.** `http://www.cprover.org/cbmc/`

Application example: Extended static checking

http://research.microsoft.com/specsharp/

Example de aplicatii: Planificare (Planning)

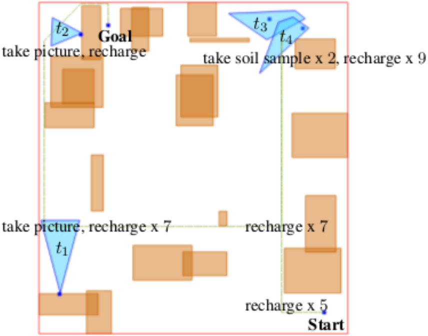


Figure 1: A GEOMETRIC ROVERS example instance, showing the starting and goal locations of the rover, areas where tasks can be performed (blue) and obstacles (orange) and a plan solving the task (green). The red box indicates the bounds of the environment.

Source: E. Scala, M. Ramirez, P. Haslum, S. Thiebaux.

Numeric planning with disjunctive global constraints via SMT.

In Proc. of ICASP'16.

Example de aplicatii: Programare (Scheduling)

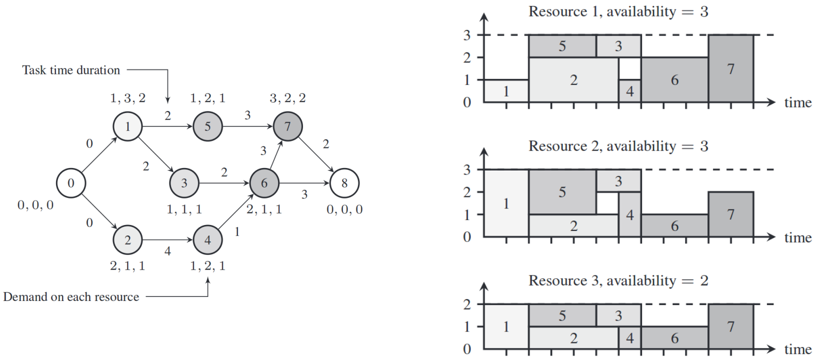


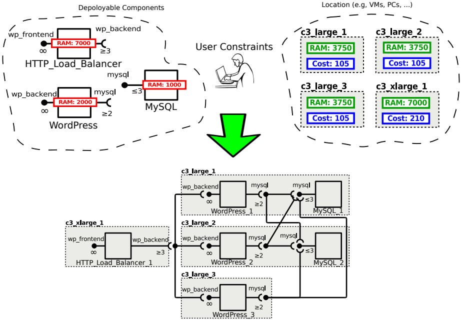
Figure 1: An example of RCPSP (Liess and Michelon 2008)

Source: C. Ansótegui, M. Bofill, M. Palahí, J. Suy, M. Villaret.

Satisfiability modulo theories: An efficient approach for the resource-constrained project scheduling problem.

Proc. of SARA'11.

Example de aplicatii: Programare (Scheduling) Application example: Optimizarea implementarii (deployment) in cloud



Source: E. Ábrahám, F. Corzilius, E. Broch Johnsen, G. Kremer, J. Mauro.

Zephyrus2: On the fly deployment optimization using SMT and CP technologies.

Submitted to SETTA'16.

Vz. si lucrarea M. Erascu, F. Micota, D. Zaharie. **Scalable Optimal Deployment in the Cloud of Component-based Applications using Optimization Modulo Theory, Mathematical Programming and Symmetry Breaking.** Submitted to Journal of Logical and Algebraic Programming