Constraint-Programmierung Vorlesung Sommersemester 2009

Johannes Waldmann, HTWK Leipzig

22. Juni 2009

Einleitung

Organisatorisches

- ›de Woche 1 Vorlesung + 1 Übung
- bungsaufgaben autotool
- rojekte (langfristig)
- rüfung ≈ Projektverteidigung

Gliederung

- ussagenlogik
  - NF-SAT-Constraints
  - PLL-Solver
  - BC, QBF
- rädikatenlogik
  - inite-Domain-Constraints
  - aive Lösungsverfahren, Konsistenzen
  - neare Gleichungen, Ungleichungen, olynomgleichungen
  - ermgleichungen, Unifikation
- odierungen nach CNF-SAT (FD, Zahlen)
- MT, DPLL(T)

Projektthemen

- lultiplikation, Faktorisierung (SAT-Encoding)
- nax,plus)-Gleichungssysteme (SMT/Ungl.)
- ttp://www.morpionsolitaire.com/
- End-)Spiele als QBF
- onstraint-Solver im Suse-Paketmanager
- uzzles von http://janko.at/ z. B.:
  http://www.janko.at/Raetsel/Nonogramme/
  http://www.janko.at/Raetsel/Zeltlager/
  Lösen und Generieren)
- ptimizer-Puzzles http://www.logic-masters.de/
  /OM2009/seiten/anleitung.php
- ernsthafte Optimierungs-Aufgaben“, Leistungsvergleich it evol. Algorithmen.

Erfüllbarkeit aussagenlogischer Formeln (SAT)

Aussagenlogik: Syntax

ısagenlogische Formel:

- lementar: Variable $v_1, \dots$, (Konstante $0, 1$)
- usammengesetzt: durch Operatoren
  - einstellig: Negation
  - zweistellig: Konjunktion, Disjunktion, Implikation, Äquivalenz

Aussagenlogik: Semantik

- Vertebereich $\mathbb{B} = \{0, 1\}$, Halbring $(\mathbb{B}, \vee, \wedge, 0, 1)$
- *elegung* ist Abbildung $b : V \to \mathbb{B}$
- *Vert* einer Formel F unter Belegung b: $\mathrm{val}(F, b)$
- venn $\mathrm{val}(F, b) = 1$, dann ist f ein *Modell* von f, chreibweise: $b \models f$
- *1odellmenge* $\mathrm{Mod}(F) = \{b \mid b \models F\}$
- ' *erfüllbar*, wenn $\mathrm{Mod}(F) \neq \emptyset$

Normalformen (DNF, CNF)

initionen:

ariable: $v_1, \ldots$

iteral: v oder $\neg v$

NF-Klausel: Konjunktion von Literalen

NF-Formel: Disjunktion von Klauseln

NF-Klausel: Disjunktion von Literalen

NF-Formel: Konjunktion von Klauseln

: Formeln F und G heißen *äquivalent*, wenn
$\mathrm{d}(F) = \mathrm{Mod}(G)$.
z: zu jeder Formel F existiert äquivalente Formel G in
F und äquivalente Formel G' in DNF.

Erfüllbarkeits-Äquivalenz

: F und G *erfüllbarkeitsäquivalent*, wenn
$\mathrm{d}(F) \neq \emptyset \iff \mathrm{Mod}(G) \neq \emptyset$.
z: es gibt einen Polynomialzeit-Algorithmus, der zu jeder
mel F eine erfüllbarkeitsäquivalente CNF-Formel G
echnet.

Tseitin-Transformation

geben F, gesucht erfüllbarkeitsäquivalentes G in CNF.
echne G mit $\mathrm{Var}(F) \subseteq \mathrm{Var}(G)$ und
$b \models F \iff \exists b' : b \subseteq b' \wedge b' \models G$.
n:

ir jeden nicht-Blatt-Teilbaum T des Syntaxbaumes von
eine zusätzliche Variable n_T einführen,
obei gelten soll: $\forall b' : \mathrm{val}(n_T, b') = \mathrm{val}(T, b')$.

alisierung:

lls (Bsp.) $F = L \vee R$, dann $n_T \leftrightarrow (n_L \vee n_r)$ als
NF-Constraintsystem
des solche System hat ≤ 8 Klauseln mit 3 Literalen, es
nd ingesamt $|F|$ solche Systeme.

Tseitin-Transportation (Übung)

ungen:

ansformiere $(x_1 \leftrightarrow x_2) \leftrightarrow (x_3 \leftrightarrow x_4)$

alb-Adder, Voll-Adder

Software zur SAT-Kodierung

Überblick

twarepaket *satchmo* (SAT encoding monad)
`p://hackage.haskell.org/cgi-bin/`
`kage-scripts/package/satchmo`

rundlagen

nonadische) Verwaltung von Variablen (State monad),
lauseln (WriterMonad) und Belegungen (ReaderMonad)

oolesche Unbekannte, Kombinatoren

elationen, Graphen

ahlen

CNF, Solver

`chmo.Data` und `Satchmo.Solve`

atentypen für Literal, Klausel, Formel

enderer für Dimacs-Format

ufruf eines externen Solvers (minisat)

arser für Ausgabe des Solvers

Automatische Hilfsvariablen

atentyp Boolean, automatisches Weiterzählen

riterMonad für CNF

eaderMonad für Resultat

```
port Satchmo.Boolean
port Satchmo.Solve
st :: SAT ( Decode Bool )
st = solve $ do
  x <- boolean -- Allokation einer Variablen
  assert [ not x ]
  return $ decode x
```

Realisierung logischer Funktionen

```
dule Satchmo.Boolean.Op where ...

d :: [ Boolean ] -> SAT Boolean
d xs = do
 <- boolean -- die Hilfsvariable
            -- lt. Tseitin-Transform
sequence $ do
   x <- xs ; return $ assert [ not y, x ]
assert $ y : map not xs
return y
```

Dekodierung mit Typklassen

ypklasse Decode

enerische Instanzen (Paare, Listen, Arrays)

```
ass Decode c a | c -> a where decode :: c ->
pe Decoder a = Reader ( Map Literal Bool ) a

stance ( Decode c a ) => Decode [c] [a] where
  decode = mapM decode
```

Relationen

```
lation :: ( Ix a, Ix b )
=> ((a,b),(a,b)) -> SAT ( Relation a b )
stance (Ix a, Ix b)
=> Decode ( Relation a b )
          ( Array (a,b) Bool ) ...
oduct :: ( Ix a , Ix b, Enum b, Ix c )
=> Relation a b -> Relation b c
-> SAT ( Relation a c )
ansitive :: ( Enum a, Ix a )
         => Relation a a -> SAT Boolean
ansitive r = do
  r2 <- product r r
  implies r2 r
```

Anwendung: Graphen

```
ight n s = do
  a <- relation ((1,1),(n,n))
  m <- atmost s $ do
         i <- indices a ; return $ a ! i
  assert [m]
  sequence_ $ do
      p <- indices a
      return $ assert $ do
          q <- indices a
          guard $ p == q || reaches p q
          return $ a!q
  return $ decode a
```

```
aches (px,py) (qx,qy) =
  5 == (px - qx)^2 + (py - qy)^2
```

SAT-Kodierung von Binärzahlen

ddition

ergleich

ubtraktion

lultiplikation

wendungen: Ramanujans Taxi, Faktorisierung

SAT: Anwendungen

SAT-Kodierungen

il SAT NP-vollständig ist, kann man jedes Problem aus nach SAT übersetzen,

. in Polynomialzeit eine Formel konstruieren, die genau ın erfüllbar ist, wenn das Problem eine Lösung hat.

spiele

idependent Set (Schach: n-Damen-Problem)

ertex Cover (Variante n-Damen-Problem)

amiltonkreis

SAT-Kodierung: Independent Set

: Graph $G = (V, E)$, Menge $M \subseteq V$ heißt unabhängig, s $\forall x, y \in M : xy \notin E$.

scheidungsproblem:

ingabe: (G, k)

rage: existiert unabhängige Menge M in G mit $|M| \geq k$?

timierungsproblem:

ingabe: G

usgabe: möglichst große unabhängige Menge in G

SAT-Kodierung: Anzahl-Constraints

$\mathrm{nt}_{\geq k}(p_1, \ldots, p_n) =$ wenigstens k der n Variablen sind ır.

allunterscheidung nach erster Variablen:

$$\begin{aligned} &\mathrm{count}_{\geq k+1}(p_1, p_2, \ldots, p_n) \\ =& p_1 \wedge \mathrm{count}_{\geq k}(p_2, \ldots, p_n) \vee \mathrm{count}_{\geq k+1}(p_2, \ldots, p_n). \end{aligned}$$

ıduktionsanfänge?

nplementierung mit Hilfsvariablen für jeden Teilausdruck entspricht Tseitin-Transformation), insgesamt $k \cdot n$ viele.

tspr. $\mathrm{count}_{=k}, \mathrm{count}_{<k}$)

SAT-Kodierung: Vertex Cover

: Graph $G = (V, E)$, Menge $M \subseteq V$ heißt
otenüberdeckung, falls $\forall x \in V \setminus M : \exists y \in M : xy \in E$.
scheidungsproblem: (G, k), Frage: … $|M| \le k$
vendung: kleinstes Vertex Cover von Damen?

SAT-Kodierung: Hamiltonkreis

: Graph $G = (V, E)$ mit $V = \{v_1, \dots, v_n\}$,
mutation $\pi : \{1, \dots, n\} \to \{1, \dots, n\}$ bestimmt
miltonkreis, wenn und
$)v_{\pi(2)} \in E, \dots, v_{\pi(n-1)}v_{\pi(n)} \in E, v_{\pi(n)}v_{\pi(1)} \in E$.
T-Kodierung: benutzt Variablen $p(i, j) \leftrightarrow \pi(i) = j$.
lche Constraints sind dafür nötig?
vendung: Rösselsprung

Suche nach symmetrischen Lösungen

s ein Problem zu groß für den Solver ist:

eitere Constraints hinzufügen …

ie effektiv die Anzahl der Variablen verkleinern.

. h.: nur nach symmetrischen Lösungen suchen.

(allgemein): f ist Symmetrie von M, falls $f(M) \sim M$.
spiel: rotationssymmetrische Aufstellungen von Figuren
Schachbrett (Vertex Cover, Hamiltonkreis).

Neue Schranke für Van der Waerdens Funktion

$r, k)$ = die größte Zahl n, für die es eine r-Färbung von
$2, \dots, n]$ gibt ohne einfarbige arithmetische Folge.
z (überhaupt nicht trivial): alle diese Zahlen sind endlich.
ige Schranken erst vor kurzer Zeit verbessert, durch
T-Kodierung, siehe Heule, M. J. H: *Improving the Odds: w Lower Bounds for Van der Waerden Numbers.* March
2008. `http://www.st.ewi.tudelft.nl/sat/`
`ides/waerden.pdf`

SAT-Solver

Überblick

ezifikation:

ingabe: eine Formel in CNF

usgabe:

eine erfüllende Belegung
oder ein Beweis für Nichterfüllbarkeit

fahren:

volutionär (Genotyp = Belegung)

kale Suche (Walksat)

PLL (Davis, Putnam, Logeman, Loveland)

Evolutionäre Algorithmen für SAT

ienotyp: Bitfolge $[x_1, \dots, x_n]$ fester Länge

hänotyp: Belegung $b = \{(v_1, x_1), \dots, (v_n, x_n)\}$

itness: z. B. Anzahl der von b erfüllten Klauseln

peratoren:

Mutation: einige Bits ändern
Kreuzung: one/two-point crossover?

blem: starke Abhängigkeit von Variablenreihenfolge

Lokale Suche (GSat, Walksat)

t Selman, Cornell University,
ry Kautz, University of Washington
`p:`
`www.cs.rochester.edu/u/kautz/walksat/`
orithmus:

eginne mit zufälliger Belegung

iederhole: ändere das Bit, das die Fitness am stärksten
rhöht

blem: lokale Optima — Lösung: Mutationen.

DPLL

vis, Putnam (1960), Logeman, Loveland (1962)
stand = partielle Belegung

ecide: eine Variable belegen

ropagate: alle Schlußfolgerungen ziehen

eispiel: Klausel $x_1 \vee x_3$, Belegung $x_1 = 0$, Folgerung:
$_3 = 1$

ei Konflikt: Backtrack/Backjump

DPLL: Heuristiken

isat http://minisat.se/

/ahl der nächsten Entscheidungsvariablen

m häufigsten in aktuellen Konflikten)

ernen von Konflikt-Klauseln (siehe nächste Folie)

orverarbeitung (Variablen und Klauseln eliminieren)
iehe Übung)

DPLL: Konfliktklauseln

jedem Konflikt:

ninimale) Ursache (d. h. Belegung) feststellen

egation davon als neue Klausel hinzufügen

unsat, Resolution

z: F in CNF nicht erfüllbar $\iff$ es gibt eine
solutions-Ableitung der leeren Klausel.

Resolutions-Schritt:

$$\frac{(x_1 \vee \ldots \vee x_m \vee y), (\neg y \vee z_1 \vee \ldots \vee z_n)}{x_1 \vee \ldots \vee x_m \vee z_1 \vee \ldots \vee z_n}$$

veis: 1. Korrektheit ($\Leftarrow$), 2. Vollständigkeit ($\Rightarrow$)

oderne SAT-Solver können solche Beweise ausgeben

s gibt nicht erfüllbare F mit (exponentiell) großen
esolutionsbeweisen (sonst wäre NP $=$ co-NP)

lls Konflikt durch Unit Progagation: (kurzer)
esolutionsbeweis $\Rightarrow$ neue Klausel

SAT: Erweiterungen

Quantifizierte Boolesche Formeln

tax: eine QBF hat die Form $\underbrace{Q_1 v_1 \ldots Q_n v_n}_{\text{Präfix}} \underbrace{M}_{\text{Matrix}}$,

ei $Q_1, \ldots, Q_n \in \{\forall, \exists\}$ und M eine aussagenlogische
mel ist (die die Variablen $v_1, \ldots, v_n$ enthalten kann)

nantik: $\mathrm{val}(M) = \mathrm{val}(M, \emptyset)$ mit

$\forall x M, b) = \mathrm{val}(M, b \cup \{(x,0)\}) \wedge \mathrm{val}(M, b \cup \{(x,1)\})$

$\exists x M, b) = \mathrm{val}(M, b \cup \{(x,0)\}) \vee \mathrm{val}(M, b \cup \{(x,1)\})$

QBF-Beispiele mit fester Quantortiefe

romatische Zahl

einstes Vertex Cover

inimales Sudoku

einster Schaltkreis

.

ema: $\exists z : P(z) \wedge \forall z' : z' < z \Rightarrow \neg P(z')$

Zweipersonenspiele

z: PSPACE $=$ APTIME (alternierend)

wendung (Projekt!): Zweipersonenspiele (mit
nomieller Länge)

$\forall w_2 \ldots \forall w_n : (s_1, w_2, \ldots, w_n)$ ist gültige Zugfolge und
sultat ist verloren für Weiß.

Komplexität

ses Problem QBF-SAT ist PSPACE-vollständig:

ingabe: eine QBF F

rage: ist $\mathrm{val}(F) = 1$

nerkung: QBF $\in$ PSPACE (alle Belegungen
chprobieren).

gabe: es ist ein (Polynomialzeit-)Verfahren anzugeben,
zu jeder polynomiell platzbeschränkten Maschine M
Eingabewort w eine QBF F konstruiert, so daß $w \to_M^*$
$\iff \models F$.

blem: F darf nur polynomiell groß sein, die Rechnungen

Lösung

veis: kodiere $F_k(p,q) \iff p \to^{\le 2^k} q$, es gibt einen Pfad
Länge $\le 2^k$ von p nach q, durch QBF der Größe $O(k)$.

$p, q) := (p = q) \vee (p \to^1 q)$

$_1(p,q) :=$

$\forall x \forall y (((x = p \wedge y = m) \vee (x = m \wedge y = q)) \Rightarrow F_k(x,y))$

wendung (Projekte!): (lange) Ableitungen (Schleifen) in
generhaltenden Ersetzungssystemen, in
schiebepuzzles (Lunar Lockout, Sokoban,...)

Satz von Savitch

z (Savitch): NPSPACE(f) $\subseteq$ DPSPACE(f^2).
nichtdeterministisch, D: deterministisch)
veis: für $M \in$ NPSPACE konstruiere Formel F wie
hin (Größe?)
ın teste $F \in$ QBF-SAT mit einer DPSPACE-Maschine.

Pseudo-Boolean Constraints

e:

lausel: $v_1 \vee \overline{v}_2 \vee v_3$

ngleichung: $v_1 + \overline{v_2} + v_3 \geq 1$

ɛudo-Boolesches Constraint (PBC) ist Ungleichung

$$c_1 v_1 + \ldots + c_n v_n \geq d,$$

Koeffizienten $c_1, \ldots, c_n, d \in \mathbb{Z}$ für Unbekannte
$. . \in \mathbb{B} = \{0, 1\}$.
C-Optimierungsaufgabe: Menge von PBC und
lfunktion $z_1 v_1 + \ldots + z_n v_n$
ɜung davon ist Belegung, die alle Constraints erfüllt und
lfunktion maximiert.

SMT (satisfiability modulo theory)

Wdhlg. Prädikatenlogik (Syntax)

ɡnatur Σ
- Relationssymbole $R_1, \ldots,$
- Funktionssymbole $f_1, \ldots,$

rm t
- Variable x
- $f(t_1, \ldots, t_k)$

rmel F
- $R(t_1, \ldots, t_k)$
- $F_1 \wedge F_2; F_1 \vee F_2; F_1 \rightarrow F_2; F_1 \leftrightarrow F_2; \neg F_1$
- $\forall x : F_1; \exists x : F_1$

nge der freien, gebundenen Variablen einer Formel
ɜchlossene (closed) Formel: keine freien Variablen

Prädikatenlogik (Semantik)

Signatur $\Sigma = (\Sigma_F, \Sigma_R)$ passende *Struktur*:

in Grundbereich M

ɹ jedem $f \in \Sigma_{F,k}$ ein $[f] : M^k \rightarrow M$

ɹ jedem $r \in \Sigma_{R,k}$ ein $[r] : M^k \rightarrow \mathbb{B}$

'egung: $b : \mathrm{Var} \rightarrow M$

iniere

/ert eines Terms t in Struktur M unter Belegung b

/ert einer Formel F in Struktur M unter Belegung b

/ert einer geschlossenen Formel F in Struktur M

Theorien

ɜorie einer Struktur = alle Formeln, die in dieser Struktur
ır sind.

spiel: Signatur $\Sigma_F = \{0_0, S_1, +_2\}, \Sigma_R = \{=_2\}$,
ɹktur: $\mathbb{N}$ mit Null, Nachfolger, Plus, Gleichheit.
ɜorie T der natürlichen Zahlen mit Addition.
mel $(\forall x : \forall y : \forall z : x + (y + z) = (x + y) + z) \in T$.

sche (algorithmische) Fragen:

rfüllbarkeit (gehört $\exists$-Abschluß von Formel F zu T?)

/ahrheit (gehört geschlossene Formel F zu T?)

Lineare Gleichungen

ɡeschränkte Syntax ($\approx$ Signatur)
me:

ummand $\rightarrow$ ganze Zahl $\cdot$ Variable

umme $\rightarrow$ Summand (+ Summand)*

meln:

ileichung $\rightarrow$ Summe $=$ Summe

ileichungssystem $\rightarrow$ Gleichung ($\wedge$ Gleichung)*

ɹktur: rationale Zahlen, übliche Operationen.
ïllbarkeit ist entscheidbar (durch Gauß-Elimination).
le Lösungsmenge ist (leer oder) affin linear.

Lineare Ungleichungen

lineare Optimierung (engl. linear programming)
me wie bei Gleichungen, aber Formeln:

ngleichung $\rightarrow$ Summe $\geq$ Summe

ngleichungssystem $\rightarrow$ Unleichung ($\wedge$ Ungleichung)*

enschaften:

rfüllbarkeit ist entscheidbar (z. B. Simplex-Algorithmus)

ösungsmenge ist Durchschnitt von Halbräumen

ösungsmenge ist (leer oder) konvex

Verfahren nach Fourier und Motzkin (I)

ısformiert ein lin. Unglsys. in ein
ïllbarkeitsäquivalentes mit einer Variablen weniger.
spiel: aus $(x + y \leq 1 \wedge 2x + 3y \geq 0)$ wird $(y \geq -2)$
ɜ ist (wie schon Gauß für Gleichungssyst.) ein Verfahren
Quantor-Elimination
$, y : x + y \leq 1 \wedge 2x + 3y \geq 0) \iff (\exists y : y \geq -2)$

Verfahren nach Fourier und Motzkin (II)

nination einer Variablen:

ähle eine Variable x,

telle alle Ungl, in denen x vorkommt, nach x um

enn alle die Form "$x \le \ldots$" haben, dann …

enn $x \le G_1 \wedge \ldots$ und $x \ge K_1 \wedge \ldots$, dann …

gen:

orrektheit

omplexität

Implementierungen

U Linear Programming Kit

`:p://www.gnu.org/software/glpk/`

`:p://www.cs.unb.ca/~bremner/docs/glpk/`
`ɔk_faq.txt`

near programming (lineare Optimierung)

nixed) integer programming (ganzzahlige lin. Opt.)

nittstellen:

ext (GNU MathProg $\subseteq$ AMPL)

PI (C, Java)

usaufgabe: installieren, ausprobieren (Text-Schnittstelle)

Fourier-Motzkin für SAT?

ederholung F-M)

: $K_1 \le x \wedge \ldots \wedge K_p \le x \wedge x \le G_1 \wedge \ldots \wedge x \le G_q$

$\Rightarrow \bigwedge_{1 \le i \le p, 1 \le j \le q} K_i \le G_j$

gleiche mit *Resolution*:

$$\begin{aligned}(A \vee x) \wedge (\neg x \vee B) &\Rightarrow (A \vee B)\\ (\neg A \to x) \wedge (x \to B) &\Rightarrow (\neg A \to B)\\ (\neg A \le x) \wedge (x \le B) &\Rightarrow (\neg A \le B)\end{aligned}$$

nination von x durch *vollständige* Resolution

$\vee x, \ldots, A_p \vee x)$ mit $(\neg x \vee B_1, \ldots, \neg x \vee B_q)$

ung: unit propagation als Spezialfall

jekt: SAT-Solver bzw. Präprozessor bauen

Mixed Integer Programming

neare Ungleichungssysteme für
lle, *ganze*, *binäre* (d.h. $\{0, 1\}$) Zahlen.

onkrete Syntax: AMPL (Robert Fourer and David M. Gay and
rian W. Kernighan, A Modeling Language for Mathematical
rogramming. Management Science 36 (1990) 519-554.
`ttp://www.ampl.com/REFS/abstracts.html#amplmod`)

olver: `http://www.ampl.com/`,
`ttp://www.gnu.org/software/glpk/`,
`ttp://scip.zib.de/`

t NP-schwer (binäre Zahlen $\to$ CNF-SAT).

Weitere Verfahren für Lin. Ungl.

ourier-Motzkin

infach, ineffizient

implex-Methode

echt einfach, meist effizient, *aber keine Polynomialzeit*

llipsoid-Methode

ompliziert, Polynomialzeit

(Un)Gl. zwischen Polynomen

veiterung der Syntax:

ummand $\to$ Zahl ($\cdot$ Variable)*

s aus der Schule bekannte „Lösen quadratischer
ichungen" ist Quantor-Elimination:

: $x^2 - x + a \le 0) \iff (a \le 1/4)$.

nehreren Variablen? — Für Struktur $\mathbb{R}$ entscheidbar

ileichungen: Gröbnerbasen

ngleichungen: Alfred Tarksi, QEPCAD,

er für $\mathbb{Z}$ nicht! (Juri Matiasevitch, Julia Robinson)

Erweiterungen

ndere Zahlen (nicht nur reelle)

ndere Strukturen (nicht nur Zahlen, (Un)gleichungen)

nehr Formeln (nicht nur $\wedge$, sondern auch $\vee$)

Anwendungsaufgabe

tivation: (Un)gleichungssysteme über

$\{-\infty\} \cup \mathbb{N}, \max, +)$ (arktischer Halbring)

$\mathbb{N} \cup \{+\infty\}, \min, \max)$ („fuzzy" Halbring)

erm $\to$ Variable | Zahl | $\max(t_1, t_2) \mid \min(t_1, t_2) \mid t_1 + t_2$

ormel $\to$ Gleichung. (Ungl. nicht nötig, warum?)

ch für reelle Zahlen) *nicht* auf lineares Ungl.sys.
uzierbar (sonst P=NP).

wendung: Terminationsbeweise mit arktischen/fuzzy
trizen

Ordnungsconstraints für Matrizen

rtersetzungssystem R über Σ,
): $R = \{aa \to aba\}$ über $\Sigma = \{a, b\}$

trix-Interpretation (Halbring D, Dimension n)
$\Sigma \to D^{n \times n}$ für Buchstaben
Wörter durch Multiplikation: $[x_1, \ldots, x_k] = [x_1] \circ \ldots \circ [x_k]$

ässig: $\forall x \in \Sigma : [x]_{1,1} \neq 0_D$
npatibel: $\forall (l, r) \in R : \forall i, j : [l]_{i,j} >_0 [r]_{i,j}$
bei $p >_0 q \iff (p = 0_D = q) \vee (p > q)$

spiel: $D = (\mathbb{N} \cup \{+\infty\}, \min, \max)$,

$$= \begin{pmatrix} 2 & +\infty & 0 \\ 1 & +\infty & 0 \\ +\infty & +\infty & +\infty \end{pmatrix}, [b] = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Lösungsansätze

Projektthemen)

mulieren als MIP, dann lösen mit glpk oder scip

MT/QFLRA, QFLIA, dann lösen mit (z. B.) Z3.2

bitblasting", SMT/QFBV oder direkt nach SAT)

ratur:

efinition, Competition (SMT/LIB, /COMP, /EXEC)
`ttp://www.smtexec.org/exec/?jobs=311`

SMT als Buch": Daniel Kroening und Oliver Strichman:
ecision Procedures—an Algorithmic Point of View,
pringer 2008. `http://www.decision-procedures.org/`

Max/Min als MIP

e: $\max(x, y) = z$ ist äquivalent zu

$$\begin{aligned} \exists b \in \{0, 1\} : \quad & x \le z \\ \wedge \quad & y \le z \\ \wedge \quad & z \le x + M \cdot b \\ \wedge \quad & z \le y + M \cdot (1 - b) \end{aligned}$$

bei M eine *genügend große* Konstante
ung: was heißt das?)

er: je kleiner M, desto besser für die Solver.

DPLL(T)

satz: für jede elementare Formel $F = P(t_1, \ldots, t_k)$
e neue boolesche Unbekannte $p_F \leftrightarrow F$,
n SAT-Problem für die p_* lösen.
en:

aiv (SAT): belegen, testen, backtrack.

PLL (SAT): partiell belegen, t., b., propagieren (, lernen)

PLL(T) (SMT): Test in Theorie T, Theorie-Propagation.

timized lazy approach)

ersichtsvortrag: Robert Nievenhuis et al. `http:`
`www.lsi.upc.edu/~roberto/RTA07-slides.pdf`

DPLL(T), Beispiel

: LRA (linear real arithmetic)
ch Tseitin-Transformation als CNF darstellen
sätzliche boolesche Variablen)
LL(T) benutzt

-Solver für Konjunktion von vollst. belegten Klauseln

:. B. Simplexverfahren)

-Propagation

:. B. $x \le y \wedge y \le z \Rightarrow x \le z$)

-Lerner

ei Nichterfüllbarkeit liefert T-Solver eine „Begründung"
kleine nicht erfüllbare Teilmenge)

Bitblasting

Idee

nstraints für ganze Zahlen

lus, minus, mal, min, max, …

oolesche Verknüpfungen

Zahlen werden „gesprengt" (blast = Explosion)
man rechnet mit ihren Bits (CNF-SAT)
ch Festlegung der Bitbreite der Zahlen)

: $[x_0, x_1, \ldots, x_{w-1}]$
ation hier: LSB ist links

Binäre Addition

$\ldots] + [y_0, \ldots] = [z_0, \ldots]$
svariablen: $[c_0, \ldots]$ = Überträge

nfang: $\mathrm{HALFADD}(x_0, y_0; z_0, c_0)$

chritt: $\forall i : \mathrm{FULLADD}(x_i, y_i, c_i; z_i, c_{i+1})$

nde: $c_w = 0$ (kein Überlauf)

alisierung:

$\mathrm{ALFADD}(x, y; z, c) \iff (z \leftrightarrow \mathrm{xor}(x, y)) \wedge (c \leftrightarrow x \wedge y)$

$\mathrm{ULLADD}(x, y, c; z, c') \iff \ldots$ (zweimal $\mathrm{HALFADD}$)

Subtraktion

ubtraktion als Umkehrung der Addition
$- y = z \iff x = z + y$

egative Zahlen als Zweierkomplement (geeignete
ehandlung von Überläufen)

Vergleiche

ßer: $x = [x_0, \ldots] > [y_0, \ldots] = y$, falls

$x_0 > y_0) \wedge (\mathrm{tail}(x) = \mathrm{tail}(y))$

der $\mathrm{tail}(x) > \mathrm{tail}(y)$

ich: $x = [x_0, \ldots] = [y_0, \ldots] = y$, falls

$| = |y| = 0$

der $(x_0 = y_0) \wedge (\mathrm{tail}(x) = \mathrm{tail}(y))$

mit $x > y$ mit linear vielen Variablen und Klauseln glich. Welches sind die besten Faktoren?

Bitshift

$< y = z \ldots$ um einen *unbekannten* Betrag y! usaufgabe.)

CNF-Kompression (Motivation)

einfachte Annahme: SAT-Solver ist schneller, wenn

eniger (Zusatz-)Variablen

eniger Klauseln

rt zu der Aufgabe:

ingabe: eine boolesche Formel F

usgabe: eine klein(st)e zu F (erfüllbarkeits)äquivalente NF.

wendungen:

ist Halbadder, Volladder

ist Addition, Multiplikation usw. für geringe Bitbreite

(Integer/Real) Difference Logic

Motivation, Definition

e Scheduling-Probleme enthalten:

ätigkeit i dauert d_i Stunden

muß beendet sein, bevor j beginnt.

führt zu Constraintsystem:

nbekannte: $t_i =$ Beginn von i

onstraints: $t_j \geq t_i + d_i$

M-LIB-Logik `QF_IDL, QF_RDL`:

lesche Kombination von

Multiplikation

$\ldots] \cdot [y_0, \ldots]$,

chulmethode: $x_0 \cdot y + \mathrm{tail}(x) \cdot y$ rekursiv

berläufe im Resultat früh erkennen, nur mit nötigen tellen rechnen

aratsuba-Multiplikation:

$+ qB)(r + sB) = pr + \underbrace{(ps + qr)}_{=t} B + qsB^2$

erechne $t = (p + q)(r + s) - pr - qs$ mit nur 3 ultiplikationen.

-Formeln für binäre Multiplikation sind sehr schwer: es Resultatbits hängt von *allen* Eingabebits ab.

Unärkodierung

ls *monotone* Bitfolge

$= [1, 1, 1, 0, 0, 0, 0, 0]$

$= [x_0, \ldots]$ mit $x_0 \geq x_1 \geq \ldots$.

bung: Min, Max, Größer.

näre Addition durch Sortiernetze (!)

= Idee der minisat-Autoren Een/Sörenssen)

CNF-Kompression (Implementierung)

gegebener Formel F bestimme Menge der *n-Implikanten* $P = \{P_1, \ldots\}$, sind *minimale* Klauseln P_i mit $F \to P_i$ nimal bezüglich Inklusion von Literalen)

n bestimme eine kleinste Teilmenge $M \subseteq P$ mit $M \to F$.

ist eine ganzzahlige lineare Optimierungsaufgabe, n durch entsprechende Constraint-Solver gelöst werden sol, scip)

Lösung von Differenz-Constraints

vie bisher) Boolesche Kombinationen werden durch PLL(T) behandelt,

er Theorie-Löser behandelt Konjunktionen von ifferenz-Konstraints.

eren Lösbarkeit ist in Polynomialzeit entscheidbar,

ilfsmittel: Graphentheorie (kürzeste Wege), schon lange ekannt (Bellman 1958, Ford 1960),

Constraint-Graphen für IDL

gegebenes IDL-System S konstruiere gerichteten
tenbewerteten Graphen G

noten i = Unbekannte t_i

ewichtete Kante $i \xrightarrow{d} j$, falls Constraint $t_i + d \geq t_j$

chte: Gewichte d können negativ sein. (wenn nicht:
blem ist trivial lösbar)
z: S lösbar $\iff$ G besitzt keinen gerichteten Kreis mit
ativem Gewicht.
lementierung: Information über Existenz eines solchen
ises fällt bei einem anderen Algorithmus mit ab.

Kürzeste Wege in Graphen

gle-source shortest paths)

ingabe:

gerichteter Graph $G = (V, E)$
Kantengewichte $w : E \to \mathbb{R}$
Startknoten $s \in V$

usgabe: Funktion $D : V \to \mathbb{R}$ mit $\forall x \in V : D(x) =$
inimales Gewicht eines Pfades von s nach x

ivalent: Eingabe ist Matrix $w : V \times V \to \mathbb{R} \cup \{+\infty\}$
(von s erreichbaren) negativen Kreisen gibt es x mit
$x) = -\infty$

Lösungsidee

ativer Algorithmus mit *Zustand* $d : V \to \mathbb{R} \cup \{+\infty\}$.
$) := 0, \forall x \neq s : d(x) := +\infty$
ile es gibt eine Kante $i \xrightarrow{w_{i,j}} j$ mit $d(i) + w_{i,j} < d(j)$
$d(j) := d(i) + w_{i,j}$
erzeit gilt die *Invariante*:

$x \in V$: es gibt einen Weg von s nach x mit Gewicht $d(x)$

$x \in V : D(x) \leq d(x)$.

bleibende Fragen:

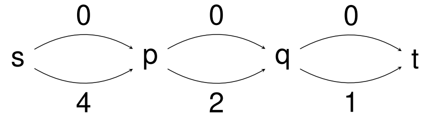
orrektheit (falls Termination)

uswahl der Kante (aus mehreren Kandidaten)

ermination, Laufzeit

Laufzeit

onentiell viele Relaxations-Schritte:



ser:

ellman-Ford (polynomiell)

r i from 1 to $|V|$: jede Kante $e \in E$ einmal

ann testen, ob alle Kanten entspannt sind.

Venn nein, dann existiert negativer Kreis. (Beweis?)

ijkstra (schneller, aber nur bei Gewichten ≥ 0 korrekt)

nwendung: Min/Max-Matrixungleichungen

Halbringoperationen in Min/Max lassen sich (trivial) in
L/IDL darstellen

Binäre Entscheidungsgraphen (ROBDDs)

Motivation

ucht ist eine Datenstruktur, die eine aussagenlogische
mel *kanonisch* repräsentiert und für die aussagenlog.
erationen *effizient* ausführbar sind.
sätze:

ormel F selbst?

lodellmenge $\mathrm{Mod}(F)$ als Menge von Belegungen?

ung: bessere Darstellung für $\mathrm{Mod}(F)$
ratur: Kroening, Struchman: Decision Procedures, 2.4.

Darstellung von Modellmengen

lnung auf Variablen festlegen: $x_1 < x_2 < \ldots < x_n$
l dann binärer Entscheidungsbaum:

lätter: 0, 1

nere Knoten t

Variable v
Kinder l, r

epräsentiert Formel $[t] = (\neg v \wedge [l]) \vee (v \wedge [r])$

Entscheidungsbäume mit Sharing

gehend von eben definiertem Baum: konstruiere DAG
richteten kreisfreien Graphen):

lätter zusammenfassen: nur zwei Blätter 0,1

lle Knoten mit gleichem (v, l, r) zusammenfassen

lle Knoten mit (v, l, r) und $l = v$ durch l ersetzen

uziertes geordnetes binäres Entscheidungsdiagramm
)BDD)
ndal E. Bryant. *Graph-Based Algorithms for Boolean
nction Manipulation*. IEEE Transactions on Computers,
5(8):677-691, 1986

Eigenschaften von ROBDDs

anonisch

rfüllbarkeit, Allgemeingültigkeit trivial

perationen? (folgende Folien)

iröße ist abhängig von Variablenreihenfolge (Beispiel)

s gibt Formeln, für die jede Variablenreihenfolge ein xponentiell großes ROBDD liefert (Beispiel)

Operationen mit ROBDDs

äre boolesche Operation f:

uf Blättern 0,1 Wert ausrechnen

uf Knoten mit gleichen Variablen: gemeins. allunterscheidung

uf Knoten mit versch. Variablen: Fallunterscheidung in einerer Variablen

Trick ist: dynamische Optimierung (d. h. von unten nach n ausrechnen und Ergebnisse merken). ultierende Laufzeit für $f(s,t)$ ist $O(|s| \cdot |t|)$.

OBDD-Anwendung: Modelle zählen

$$|\operatorname{Mod}(0)| = 0, |\operatorname{Mod}(1)| = 1,$$
$$|\operatorname{Mod}(v,l,r)| = |\operatorname{Mod}(l)| + |\operatorname{Mod}(r)|$$

se Zahlen bottom-up dranschreiben: Linearzeit

Konsistenz-Begriffe

Finite-Domain-Constraints (FD)

inition

rädikatenlogische Formel über mehrsortiger Signatur,

ir jede Sorte eine endliche Menge

spiele

agisches Quadrat, Sudoku

össelsprung, n-Damen-Problem

raphenfärbungen

AT

FD-Lösungsverfahren

nzip (vgl. DPLL für SAT)

ntscheiden:

ine Variable auswählen, für diese alle möglichen elegungen ausprobieren

ropagieren:

lenge der möglichen Belegungen anderer Variablen erkleinern

che Variable wird Entscheidungsvariable?

ktivität

Vertebereich

Konsistenzen (Überblick)

n betrachtet jeweils

onstraints (log. Formeln) $C(\ldots, x, \ldots)$

ereiche $x \in \{e_1, \ldots, e_n\} = E$

nstraint-System heißt …-konsistent, wenn ausgewählte nstraints und Bereiche zusammenpassen.

nn es nicht paßt: Bereiche durch gewisse Regeln kleinern (bis konsistent oder offensichtlich unlösbar)

notenkonsistenz

antenkonsistenz

fadkonsistenz

Knotenkonsistenz

Constraintsystem heißt *knotenkonsistent*, wenn für jede iable x gilt:

der Wert aus dem Bereich von x …

rfüllt alle Constraints, in denen nur x vorkommt.

Kantenkonsistenz

Constraintsystem heißt *kantenkonsistent*, wenn für alle iablenpaare x, y gilt:

u jedem Wert a aus dem Bereich von x …

ibt es einen Wert b aus dem Bereich von y,

o daß (a,b) alle Constraints erfüllt, in denen nur x und y orkommen.

chte: auch für y, x

Pfadkonsistenz

Constraintsystem heißt *pfadkonsistent*, wenn für alle iablen x, y, z gilt:

$C_{x,y} \circ C_{y,z} \supseteq C_{x,z}$

bei $C_{x,y} =$ Menge aller erfüllenden Belegungen ($=$ ordnete Paare) von Constraints, in denen nur x, y kommen.

k-Konsistenz

e partielle Belegung b von Variablen eines CS heißt *nsistent*, wenn b alle Constraints erfüllt, in denen nur iablen aus $\operatorname{dom} b$ vorkommen.

CS ist k-konsistent, wenn jede konsistente $-1)$-Belegung zu einer konsistenten k-Belegung gesetzt werden kann.

nauer: ...

ziehungen zu Knoten-, Kanten-, Pfadkonsistenz.

Konsistenz unter Kodierungen

spiel: FD-CS für Zahlen und SAT-CS für binäre dierung.

st möglich, daß das Original inkonsistent ist, aber die dierung konsistent.

spiel: $x \in [0, 1, \ldots, 5]$ unär und binär

Gleichheits-Constraints

(Dis)Equality Constraints (Bsp, Def)

$$\begin{aligned} & (a_0 = b_0 \wedge b_0 = a_1 \vee a_0 = c_0 \wedge c_0 = a_1) \wedge \ldots \\ \wedge \quad & (a_n = b_n \wedge b_n = a_{n+1} \vee a_n = c_n \wedge c_n = a_{n+1}) \\ \wedge \quad & a_{n+1} \neq a_0 \end{aligned}$$

e: Strichman, Rozanov, SMT Workshop 2007,

//www.lsi.upc.edu/~oliveras/espai/smtSlides/ofer.ppt,

//www.smtexec.org/exec/benchmarkDetail.php?benchmark=661781)

mel $\exists x_1, \ldots, x_n : M$ mit M in negationstechnischer malform über Signatur:

rädikatsymbole $=$ und $\neq$, keine Funktionssymbole

Gleichheitsgraph (equality graph)

noten: Variablen

anten xy, falls $x = y$ oder $x \neq y$ in M vorkommt Gleichheitskante, Ungleichheitskante)

gische Verknüpfungen werden ignoriert!)

dersspruchskreis:

eschlossene Kantenfolge

enau eine Ungleichheitskante

infach (simple): kein Knoten doppelt

s die Kanten eines solchen Kreise mit *und* verknüpft d, dann ist die Formel nicht erfüllbar.

Vereinfachen von Formeln

z: Falls M eine Teilformel T (also $x = y$ oder $x \neq y$) hält, die auf keinem Widerspruchskreis vorkommt, n ist $M' := M[T/\text{true}]$ erfüllbarkeitsäquivalent zu M.

weis: eine Richtung ist trivial, für die andere: konstruiere erfüllender Belegung von M' eine erfüllende Belegung M.

orithmus: solange wie möglich

ine solche Ersetzung ausführen

ormel vereinfachen

Reduktion zu Aussagenlogik (I)

lan: jede elementater Formel ($x = y$) durch eine oolesche Variable ersetzen, dann SAT-Solver nwenden.

Widerspruchskreise sind auszuschließen: in jedem Kreis arf nicht genau eine Kante falsch sein.

adurch wird die Transitivität der Gleichheitsrelation usgedrückt.

s gibt aber exponentiell viele Kreise.

Reduktion zu Aussagenlogik (II)

z (Bryant und Velev, CAV-12, LNCS 1855, 2000): genügt, Transitivitäts-Constraints für sehnenlose Kreise zuzufügen.

z (Graphentheorie/Folklore): edem Graphen G kann man in Polynomialzeit einen rdalen Obergraphen G' konstruieren (durch Hinzufügen Kanten).

ph G heißt *chordal*, wenn er keinen sehnenlosen Kreis er Länge ≥ 4 enthält.

sicht: chordal $\neq$ trianguliert, Beispiel.

orithmus: wiederholt: einen Knoten entfernen,

Ausflug Graphentheorie

ordale Graphen G sind *perfekt*: für jeden induzierten
graphen $H \subseteq G$ ist die *Cliquenzahl* $\omega(H)$ ist gleich der
omatischen Zahl $\chi(H)$.

iter Klassen perfekter Graphen sind:

ipartite Graphen, z. B. Bäume

olit-Graphen, Intervallgraphen

C_5 ist nicht perfekt)

ong Perfect Graph Theorem (Vermutung: Claude Berge
60, Beweis: Maria Chudnovsky und Paul Seymor 2006):
perfekt $\iff$ G enthält kein C_{2k+1} oder $\overline{C_{2k+1}}$ für $k \geq 2$.
`p://users.encs.concordia.ca/~chvatal/`
`rfect/spgt.html`

Kleine Welt

Gleichheits-Constraints mit n Variablen $x_1, \ldots, x_n$:
hier gezeigte Kodierung benutzt n^2 boolesche Variablen
$\iff (x_i = x_j)$

kann man reduzieren:

nn ein solches Gleichheits-System erfüllbar ist, dann
sitzt es auch ein Modell mit $\leq n$ Elementen.
weis-Idee: schlimmstenfalls sind alle Variablenwerte
schieden)

n Wert jeder Variablen kann man also mit $\log n$ Bits
lieren.

veiterung: man kann für jede Variable einen passenden
tl. kleineren) Bereich bestimmen.

Uninterpretierte Funktionen (UF)

Motivation, Definition

hrheit/Erfüllbarkeit hängt von Theorie (d.h.
erpretation der Funktionssymbole) ab, Bsp:

$$\forall a, b : a \circ b = b \circ a$$

für Zahlen, aber nicht für Matrizen. Aber

$$x, y, z : g(g(x)) = x \wedge g(h(x)) = x) \rightarrow (\forall x : h(g(x)) = x)$$

allgemeingültig in *jeder* Theorie,
l es in der Theorie der *freien Termalgebra*
emeingültig ist.

Terme

ignatur Σ:

lenge von Funktionssymbolen mit Stelligkeiten

$\in \mathrm{Term}(\Sigma)$: geordneter gerichteter Baum mit zu Σ
assender Knotenbeschriftung

-Algebra:

Trägermenge D
zu jedem k-stelligen Symbol $f \in \Sigma$
eine Funktion $[f] : D^k \rightarrow D$.

amit wird jedem $t \in \mathrm{Term}(\Sigma)$ ein Wert $[t] \in D$
ugeordnet

$\mathrm{erm}(\Sigma)$ ist selbst eine Σ-Algebra

Gleichheit von Termen

Relation $=$ wird beschrieben durch das Axiom

$$t_1 = s_1) \wedge \ldots \wedge (t_k = s_k) \rightarrow f(t_1, \ldots, t_k) = f(s_1, \ldots, s_k)$$

SMT-Sprechweise: *functional consistency*)

efinition: eine Σ-Algebra heißt *frei*, wenn keine weiteren
leichheiten gelten.

eispiel: jede Termalgebra ist frei.

iegen-Beispiel: $\Sigma = \{+, 1\}, D = \mathbb{N}$ ist nicht frei:
$1 + 1) + 1 = 1 + (1 + 1)$.

Logik und Unentscheidbarkeit (I)

: das Gültigkeitsproblem der Prädikatenlogik:

ingabe: eine prädikatenlogische Formel F über Σ,

rage: ist F in $\mathrm{Term}(\Sigma)$ allgemeingültig

spiel: Σ: zwei einstellige Funktionsymbole f, g,
nullstelliges Funktionssymbol a,
zweistelliges Prädikatsymbol R.

$$\forall x, y : R(x, y) \rightarrow \begin{pmatrix} & R(f(x), g(f(f(y)))) \\ \wedge & R(g(x), f(y)) \\ \wedge & R(g(f(f(x))), g(y)) \end{pmatrix}$$
$$\wedge \quad R(a, a) \wedge \exists z : R(g(z), g(z))$$

Logik und Unentscheidbarkeit (II)

z: Das Gültigkeitsproblem der Prädikatenlogik ist
entscheidbar.
rt Gödel, Alan Turing.
`p://thocp.net/biographies/papers/turing_`
`computablenumbers_1936.pdf`)

weis:

alteproblem für TM ist unentscheidbar

alteproblem $\leq$ Postsches Korrespondenzproblem (PCP)

CP $\leq$ Gültigkeitsproblem

wendet *Reduktion* $A \leq B :\iff$

Entscheidbare Fälle

durch Einschränkung der Position der Quantoren.
SMT-LIB: nur außen existentiell.

$$\exists f, g, x, y : \neg((x = y) \rightarrow (f(f(g(x))) = f(f(g(y))))$$

```
enchmark check  :logic QF_UF
xtrafuns ((f U U)(g U U)(x U)(y U))
ormula (not (implies (= x y)
                (= (f(f(g x))) (f(f(g y))))
```

nt erfüllbar, d. h. das Gegenteil ist allgemeingültig:

$$\forall f, g, x, y : ((x = y) \rightarrow (f(f(g(x))) = f(f(g(y))))$$

Ackermann-Transformation

von Formel F in QF_UF nach Formel F' in *equalitiy logic* QF_UF mit nur nullstelligen Symbolen)

rsetze jeden vorkommenden Teilterm F_i durch ein neues ullstelliges Symbol f_i

ige *functional consistency constraints* hinzu:

ir alle $F_i = g(F_{i1}, \ldots, F_{ik}), F_j = g(F_{j1}, \ldots, F_{jk})$ mit $i \neq j$:

$f_{i1} = f_{j1} \wedge \ldots \wedge f_{ik} = f_{jk}) \rightarrow f_i = f_j$

z: F erfüllbar $\iff$ F' erfüllbar.

Kombination von Theorien

Motivation

oening/Strichman: Kapitel 10)

ineare Arithmetik + uninterpretierte Funktionen:

$x_2 \geq x_1) \wedge (x_1 - x_3 \geq x_2) \wedge (x_3 \geq 0) \wedge f(f(x_1) - f(x_2)) \neq (x_3)$

itfolgen + uninterpretierte Funktionen:

$(a[32], b[1]) = f(b[32], a[1]) \wedge a[32] = b[32]$

rrays + lineare Arithmetik

$= v\{i := e\}[j] \wedge y = v[j] \wedge x > ex > y$

Definition

dhlg) Signatur Σ, Theorie T, Gültigkeit $T \models \phi$

nbination von Theorien:

heorie T_1 über Σ_1, Theorie T_2 über Σ_2.

heorie $T_1 \oplus T_2$ über $\Sigma_1 \cup \Sigma_2$

gabenstellung:

egeben: Entscheidungsverfahren (Constraint-Solver) für T_1, T_2 (einzeln)

esucht: … für $T_1 \oplus T_2$

Konvexe Theorien

e Theorie T heißt *konvex*, wenn für jede Formel ϕ, Zahl Variablen $x_1, \ldots, x_n, y_1, \ldots, y_n$:

us $T \models (\phi \rightarrow (x_1 = y_1) \vee \ldots \vee (x_n = y_n))$

olgt: es gibt ein i mit $T \models (\phi \rightarrow (x_i = y_i))$

nvex oder nicht?

neare Arithmetik über $\mathbb{R}$

neare Arithmetik über $\mathbb{Z}$

onjunktionen von (Un)gleichungen

Nelson-Oppen (I)

dhlg) Formel/Ausdruck

ification (Reinigung):

urch Einführen neuer Variablen:

lle atomaren Formeln enthalten nur Ausdrücke *einer* heorie

spiel:

orher: $\phi := x_1 \leq f(x_1)$

achher: $\phi' := x_1 \leq a \wedge a = f(x_1)$

Allg. $\phi \iff \exists a, \ldots : \phi'$

ı. ϕ erfüllbar $\iff$ ϕ' erfüllbar.

Nelson-Oppen für konvexe Theorien

entscheidbare Theorien $T_1, \ldots, T_n$:

ingabe: gereinigte Formel $\phi = \phi_1 \wedge \ldots \wedge \phi_n$

enn ein ϕ_i nicht erfüllbar, dann ist ϕ nicht erfüllbar

enn $T_i \models (\phi_i \rightarrow x_i = y_i)$, dann Gleichung $x_i = y_i$ zu llen ϕ_j hinzufügen,…

is sich nichts mehr ändert, dann ϕ erfüllbar

ispiele)

weis)

Nelson-Oppen für nicht konvexe Theorien

s $T_i \models (\phi_i \rightarrow (x_1 = y_1 \vee \ldots x_k = y_k))$,

n *split*: für $1 \leq j \leq k$ betrachte $\phi \wedge (x_j = y_j)$

ispiel)

weis)

Programme mit Arrays

Definition

MTLIB: QF_A)

ignatur mit Sorten: Array, Index, Element

ymbole: Gleichheit,

elect : Array $\times$ Index $\rightarrow$ Element,

tore : Array $\times$ Index $\times$ Element $\rightarrow$ Array

xiom: $\forall a \in$ Array$, i, j \in$ Index$, e \in$ Element :

$\mathrm{elect}(\mathrm{store}(a, i, e)) = \ldots$

n McCarthy and James Painter: *Correctness of a mpiler for Arithmetic Expressions*, AMS 1967, http:

Store entfernen

ɟeben eine ϕ Formel mit Teilausdruck $\mathrm{store}(a, i, e)$,
etze diesen durch a' und füge Constraints hinzu:
$\mathrm{ct}(a', i) = e \wedge \forall j \neq i : \mathrm{select}(a', j) = \mathrm{select}(a', i)$.
es sich um quantorenfreie Formeln handelt: j läuft über
Index-Ausdrücke, die in der Formel vorkommen.

Spezialfälle

ch beschriebene Transformation kann man Array-Logik
uzieren auf Kombination von

ıdex-Logik (z. B. lineare Arithmetik)

lement-Logik

ninterpretierte Funktionen.

ɜen mit Nelson-Oppen möglich.
ıliche Rechnungen finden (seit vielen Jahrzehnten!) im
ıleifen-Optimiere von `gcc` statt (Beispiel).
Array-Logik kann man beweisen, daß der Optimierer
ht hat.