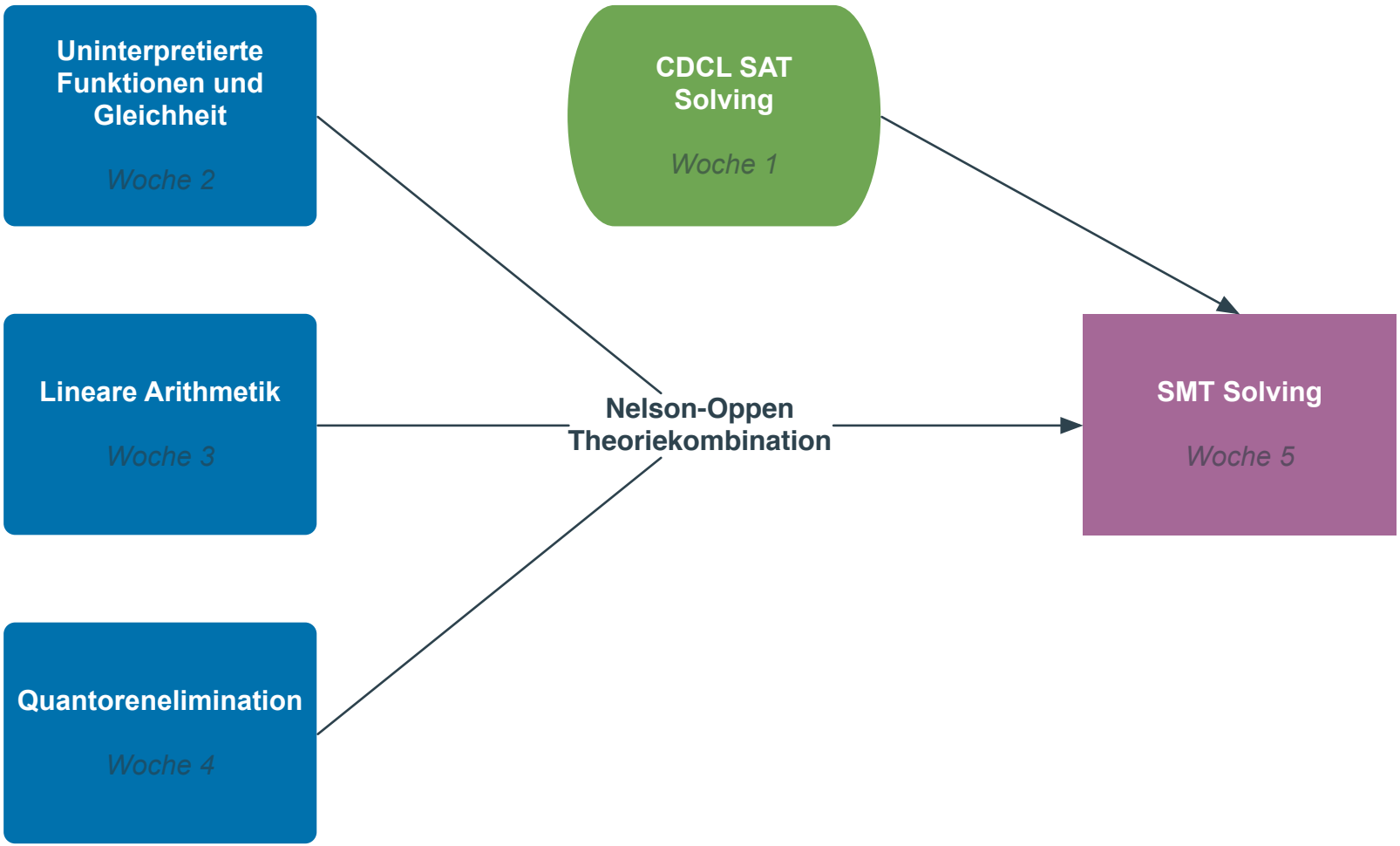
Automatisches Beweisen—Vertiefung
SMT Solving
und das DPLL($\mathcal{T}$) Framework

Christoph Zengler

Arbeitsbereich Symbolisches Rechnen
Prof. Dr. Wolfgang Küchlin
Universität Tübingen

13. Dezember 2011

Wir erinnern uns...



Motivation

- Viele wissenschaftlich und industriell interessante Probleme können in SAT codiert und mit SAT Solvern gelöst werden
- Codierung nach SAT oft problemlos möglich (Hardware, Software,...)
- Problem: Codierungen nach SAT werden oft zu groß

⚠ Beispiel: Lineare Arithmetik

- Um einen arithmetischen Ausdruck zu codieren muss man zuerst ein Addierwerk, und darauf aufbauend einen Multiplizierer in Aussagenlogik modellieren
- Codierung eines n-Bit Multiplizierer ist hart:

n	Anzahl Variablen	Anzahl Klauseln
8	313	1001
16	1265	4177
24	2857	9529
32	5089	17057
64	20417	68929

Motivation

- Für viele Theorien sind bereits Entscheidungsverfahren bekannt, die mehr Domänenwissen einbeziehen als eine Codierung nach SAT
  - **Gleichheitslogik:** Siehe Vorlesung
  - **Uninterpretierte Funktionen:** Siehe Vorlesung
  - **Lineare Arithmetik:** Siehe Vorlesung
  - **Bit Vektoren:** Bit-Flattening
  - **Arrays:** Übersetzung zu uninterpretierten Funktionen
  - **Zeigerlogik:** Modellierung des Speichers als Array
  - **Quantifizierte Formeln:** Quantorenelimination
- Warum also nicht einfach die jeweilige Logik benutzen und Solver für diese Tools verwenden? $\Rightarrow$ **SAT Modulo Theories (SMT)**
- Problem: Kombination von Theorien

Beispiel (Kombination von Theorien)

$$\underbrace{g(a) = x \wedge (f(g(a)) \neq f(c) \vee g(a) = d)}_{\text{Uninterpretierte Funktionen und Gleichungslogik}} \wedge \underbrace{a < d \wedge f(a) \leqslant c}_{\text{Lineare Arithmetik}}$$

Grundidee für SMT—1

- Löse AL-Skelett der Formel mit SAT Solver
- Skelett UNSAT $\Rightarrow$ original Formel UNSAT
- Sonst: Verwende erfüllende Belegung und schicke die Konjunktion der jeweiligen atomaren Formeln an die jeweiligen Theorie Solver
- Versuche aus fehlgeschlagenen Versuchen zu lernen

Beispiel (SMT Solving)

$$g(a) = c \wedge (f(g(a)) \neq f(c) \vee g(a) = d) \wedge c \neq d$$

- Ersetze atomare Formeln durch neue aussagenlogische Formeln:

$$P_1 \wedge (\neg P_2 \vee P_3) \wedge \neg P_4$$

- SAT Solver liefert Modell $P_1, \neg P_2, \neg P_4$ zurück
- Konjunktion atomarer Formeln wird an Theory Solver geschickt:

$$g(a) = c \wedge f(g(a)) \neq f(c) \wedge c \neq d$$

Grundidee für SMT—2

Beispiel (SMT Solving)

- Formel: $g(a) = c \wedge (f(g(a)) \neq f(c) \vee g(a) = d) \wedge c \neq d$
- Skelett: $P_1 \wedge (\neg P_2 \vee P_3) \wedge \neg P_4$
- 1. Modell: $P_1, \neg P_2, \neg P_4$

- Theory Solver liefert UNSAT
- Dieses Modell wird in Zukunft geblockt (blocking clause)
- Schicke an SAT Solver: $\{(P_1), (\neg P_2, P_3), (\neg P_4), (\neg P_1, P_2, P_4)\}$
- SAT Solver liefert Modell $P_1, P_2, P_3, \neg P_4$
- Theory Solver liefert UNSAT
- Neue Formel:
  $\{(P_1), (\neg P_2, P_3), (\neg P_4), (\neg P_1, P_2, P_4), (\neg P_1, \neg P_2, \neg P_3, P_4)\}$
- SAT Solver liefert UNSAT $\Rightarrow$ Formel ist UNSAT

Syntax

Signatur

Definition (Signatur)

Signatur Σ: Menge an Prädikats- und Funktionssymbolen

- Jedes Symbol hat eine Stelligkeit (Arität)
- Σ^{P} bzw. Σ^{F} Menge der Prädikats- bzw. Funktionssymbole
- 0-stellige Funktionssymbole: Konstanten
- 0-stellige Prädikatessymbole: Aussagenlogische Konstanten

Notation:

- $\mathrm{a}, \mathrm{b}, \mathrm{c}$: Konstanten
- A, B: Aussagenlogische Konstanten
- f, g: nicht-konstante Symbole aus Σ^{F}
- P, Q: nicht-konstante Symbole aus Σ^{P}

Syntax

EBNF-Grammatik für SMT Formeln

Term	$=$	*Konstante*	aus Σ^{F} mit Arität 0
	$\mid$	*Funktionssymbol*({Term})	Funktion
Formel	$=$	*Aussagenlogische Konstante*	aus Σ^{P} mit Arität 0
	$\mid$	*Prädikatssymbol*({Term})	Prädikat
	$\mid$	Term = Term	Gleichung
	$\mid$	$\top \mid \bot \mid \neg$Formel	
	$\mid$	Formel $\vee$ Formel	
	$\mid$	Formel $\wedge$ Formel	
	$\mid$	Formel $\rightarrow$ Formel	
	$\mid$	Formel $\leftrightarrow$ Formel	

- Atomare Formel: AL Konstanten, Prädikate, Gleichungen, $\bot$ und $\top$
- Literal: Atomare Formel oder deren Negation (Notation: $\mathfrak{l}$)

? Warum keine Variablen?

Aus technischen Gründen werden Variablen wie Konstanten behandelt. Der Σ-Term $\mathrm{x} < \mathrm{y} + 1$ ist variablenfrei und x und y werden als Konstanten zu Σ hinzugefügt.

Semantik—1

- Im Gegensatz zur herkömmlichen FOL brauchen wir keine Belegung β, da keine Variablen
- Interpretation $\mathcal{M}$: Universum $\mathcal{D}$ & Interpretation der Prädikats- und Funktionssymbole

Algorithmus: $\texttt{holds}(\mathcal{M}, \psi)$

Eingabe: Interpretation $\mathcal{M}$, SMT Formel ψ
Ausgabe: Evaluation von ψ unter $\mathcal{M}$

$\texttt{holds}(\mathcal{M}, \psi) = \psi$ *match*

$\top \rightsquigarrow \texttt{true}$

$\mid \bot \rightsquigarrow \texttt{false}$

$\mid P(t_1, \ldots, t_n) \rightsquigarrow P_{\mathcal{M}}(\texttt{termval}(\mathcal{M}, t_1), \ldots, \texttt{termval}(\mathcal{M}, t_n))$

$\mid t_1 = t_2 \rightsquigarrow \texttt{termval}(\mathcal{M}, t_1) = \texttt{termval}(\mathcal{M}, t_2)$

$\mid \neg\varphi \rightsquigarrow$ *if* $\texttt{holds}(\mathcal{M}, \varphi)$ *then* $\texttt{false}$ *else* $\texttt{true}$

$\mid \ldots$

Semantik—2

Erfüllbarkeitsbegriff

- Eine Σ-Interpretation $\mathcal{M}$ erfüllt (bzw. falsifiziert) eine Σ-Formel φ gdw. $\texttt{holds}(\mathcal{M}, \mathrm{phi}) = \texttt{true}$ (bzw. $= \texttt{false}$)
- Erfüllt $\mathcal{M}$ φ, so ist $\mathcal{M}$ ein Modell von φ
- **In SMT:** Keine beliebige Interpretation, sondern Interpretation, die zu einer bestimmten Theorie $\mathcal{T}$ gehört

Definition (Σ-Theorie)

Ein oder mehrere (möglicherweise unendlich viele) Σ-Interpretationen

Beispiel (Theorien)

- Für $\Sigma_{\mathbb{Z}} = \{0, 1, +, -, \leqslant\}$ ist $\mathcal{T}_{\mathbb{Z}}$ die Menge aller Interpretationen, die diese Symbole in herkömmlicher Weise über $\mathbb{Z}$ interpretieren
- Für eine beliebige Signatur Σ ist $\mathcal{T}_{=}$ die Menge aller Interpretationen (Gleichheitslogik mit uninterpretierten Funktionen)

Semantik—3

Definition ($\mathcal{T}$-Erfüllbarkeit)

Eine variablenfreie Σ-Formel φ ist $\mathcal{T}$-erfüllbar, gdw. es in $\mathcal{T}$ ein Modell für φ gibt.

Definition ($\mathcal{T}$-Folgerung)

Eine Menge Γ von variablenfreien Σ-Formeln $\mathcal{T}$-folgert eine Formel φ, $\Gamma \models_{\mathcal{T}} \varphi$, gdw. jede Interpretation aus $\mathcal{T}$, die ein Modell für alle Formeln in Γ ist, auch ein Modell für φ ist.

Definition ($\mathcal{T}$-Konsistenz)

Γ ist $\mathcal{T}$-konsistent gdw. $\Gamma \not\models_{\mathcal{T}} \bot$

Definition ($\mathcal{T}$-Validität)

φ ist $\mathcal{T}$-valide gdw. jede Interpretation in $\mathcal{T}$ ein Modell für φ ist

Uninterpretierte Symbole

- Wir wollen uninterpretierte Symbole zulassen
  - uninterpretierte Konstanten: Ersetzen Variablen
  - uninterpretierte Aussagenlogische Konstanten: Ersetzen Teilformeln

Formal betrachten wir nicht $\mathcal{T}$, sondern eine Erweiterung $\mathcal{T}'$

Definition (Erweiterung $\mathcal{T}'$)

- Σ' beliebige Signatur, die Σ enthält

Eine Erweiterung $\mathcal{M}'$ zu Σ' einer Σ-Interpretation $\mathcal{M}$ ist eine Σ'-Interpretation mit

- dem selbem Universum wie $\mathcal{M}$ und
- alle Symbole aus Σ werden in Σ' gleich interpretiert.

$\mathcal{T}'$ ist die Menge aller möglichen Erweiterungen der Interpretationen aus $\mathcal{T}$ zu Σ'.

Was hat es mit Erweiterungen auf sich?

Idee!

- Wir haben keine Variablen, nur Konstanten
- Für eine Theorie fixieren wir bestimmte Symbole
- Konstanten können uninterpretatiert bleiben

Erweiterung betrachtet nun alle möglichen Interpretationen dieser Konstanten

Wir sprechen jedoch weiterhin von $\mathcal{T}$-erfüllbar, $\mathcal{T}$-folgern, usw., haben jedoch im Kopf, dass wir eigentlich von der Erweiterung $\mathcal{T}'$ sprechen

Problemstellung: ground $\mathcal{T}$-satisfiability problem

Gegeben eine Σ-Theorie $\mathcal{T}$. Ist eine variablenfreie Formel über einer beliebigen Erweiterung von Σ mit uninterpretierten Konstanten $\mathcal{T}$-erfüllbar?

Bemerkung: φ ist $\mathcal{T}$-unerfüllbar, gdw. $\neg\varphi$ $\mathcal{T}$-valide ist.

Kombinierte Theorien—1

Kombination von zwei oder mehr Theorien $\mathcal{T}_1$ und $\mathcal{T}_2$ in einer Formel:

- Theorien über Axiome definiert? $\Rightarrow$ neue Theorie Vereinigung der beiden Axiommengen
- Besitzen die beiden Signaturen von $\mathcal{T}_1$ und $\mathcal{T}_2$ gemeinsame Symbole?
  - Haben diese Symbole nicht die selbe Bedeutung (selbe Relation bzgl. des Universums), muss eines der beiden umbenannt werden

Bei uns jedoch: Theorie ist eine Menge von Interpretationen

Kombinierte Theorien—2

Bei uns jedoch: Theorie ist eine Menge von Interpretationen

Definition (Reduktum)

Eine Σ-Interpretation $\mathcal{M}$ ist das Σ-Reduktum einer Σ'-Interpretation $\mathcal{M}'$ mit $\Sigma \subseteq \Sigma'$, wenn

- $\mathcal{M}$ das selbe Universum hat wie $\mathcal{M}'$ und
- $\mathcal{M}$ die Symbole exakt wie $\mathcal{M}'$ interpretiert

Definition (Kombination von Theorien)

Die Kombination $\mathcal{T}_1 \oplus \mathcal{T}_2$ von $\mathcal{T}_1$ und $\mathcal{T}_2$ ist die Menge aller $(\Sigma_1 \cup \Sigma_2)$-Interpretationen $\mathcal{M}$, deren

- Σ_1-Reduktum isomorph ist zu einer Interpretation aus $\mathcal{T}_1$ und
- Σ_2-Reduktum isomorph ist zu einer Interpretation aus $\mathcal{T}_2$.

Abstraktion

Assoziiere mit jeder Signatur Σ eine Signatur Ω, die enhält:

- aussagenlogische Konstanten von Σ
- Menge an neuen aussagenlogischen Symbolen mit der selben Kardinalität wie die Menge der variablenfreien Σ-Atome

Definition (Aussagenlogisches Skelett)

Bijektion $\mathcal{T}2\mathcal{B}$ zwischen den variablenfreien Σ-Formeln und den aussagenlogischen Formeln über Ω

- Jede aussagenlogische Konstante aus Σ wird auf sich selbst abgebildet
- Alle atomaren Formeln werden auf neue Symbole in Ω abgebildet

Definition (Refinement)

Inverse Abbildung von $\mathcal{T}2\mathcal{B}$ ist $\mathcal{B}2\mathcal{T}$.

Abstraktion

Notationen & Beispiele

Notation

- φ^{p} anstelle von $\mathcal{T}2\mathcal{B}(\varphi)$
- Für Menge Γ an Σ-Formeln: $\Gamma^{\mathrm{p}} = \{\varphi^{\mathrm{p}} \mid \varphi \in \Gamma\}$
- Eine Σ-Formel φ ist aussagenlogisch unerfüllbar, wenn $\varphi^{\mathrm{p}} \models \bot$

Beispiel (Aussagenlogisches Skelett)

- $\varphi = A \wedge a = c \vee P(a, b) \wedge Q(c)$
- Signatur $\Sigma = \{A^{(0)}, P^{(2)}, Q^{(1)}, a^{(0)}, b^{(0)}, c^{(0)}\}$
- $\Omega = \{A^{(0)}, R_1^{(0)}, R_2^{(0)}, R_3^{(0)}\}$

$$\mathcal{T}2\mathcal{B}(\varphi) = \varphi^{\mathrm{p}} = A \wedge R_1 \vee R_2 \wedge R_3$$

$$\mathcal{B}2\mathcal{T}(\varphi^{\mathrm{p}}) = \varphi$$

Interessante Theorien—1

Was wir bereits kennen

- Gleichheitslogik und uninterpretierte Funktionen
- Lineare Arithmetik

Was es noch gibt

Restriktionen der Arithmetik

- **Differenzlogik**: Jedes Atom ist von der Form $\mathrm{a} - \mathrm{b} \bowtie \mathrm{t}$ mit
  - a und b uninterpretierte Konstanten
  - $\bowtie \in \{=, \leqslant\}$
  - t Ganzzahl
- **Unit Two Variables per Inequality**: Erlaubt neben obigen Formeln auch noch $\mathrm{a} + \mathrm{b} \bowtie \mathrm{t}$

Interessante Theorien—2

Arrays

- Signatur $\Sigma = (\texttt{read}, \texttt{write})$
- Array a
  - $\texttt{read}(a, i)$: Wert von a am Index i
  - $\texttt{write}(a, i, v)$: Array, dass identisch zu a ist, nur dass an Index i der Wert v steht

Formale Axiome

1. $\forall a\ \forall i\ \forall v\ (\texttt{read}(\texttt{write}(a, i, v), i) = v)$
2. $\forall a\ \forall i\ \forall j\ \forall v\ (i \neq j \rightarrow \texttt{read}(\texttt{write}(a, i, v), j) = \texttt{read}(a, j))$

- Theorie $\mathcal{T}_A$ umfasst alle Modelle dieser Axiome

Bit-Vektoren

- Vektoren von Bits mit fester Länge n
- Operatoren: Extraktion, Konkatenation, Bitweise Operatoren, ...

Eager vs. Lazy Encodings

Eager Approach

- **Idee:** Übersetze gesamtes Problem in eine erfüllbarkeitsäquivalente Formel in Aussagenlogik und benutze SAT Solver
- **Warum „eager“:** Sämtliche Theorieinformation wird von Anfang an verwendet
- **Vor/Nachteile:**
  - \+ Fortschritte in SAT direkt nutzbar (Verwende besten SAT Solver)
  - \- Schwierige Kodierung einiger Theorien in AL

Lazy Approach

- **Idee:** Rufe Theory Solver erst auf, wenn er gebraucht wird
- **Warum „lazy“:** Theorieinformation wir erst benutzt, wenn der jeweilige Theory Solver aufgerufen wird
- **Vor/Nachteile:**
  - \+ Modular und Flexibel (Theory Solver können gepluggt werden)
  - \- Die Suche wird nicht durch die Theorieinformation geleitet

Eager Approach—Beispiel

Beispiel (Eager Encoding)

$$f(a) = f(b) \wedge f(b) \neq f(c)$$

1. Ersetze Funktionen und Prädikate durch Konstanten (z.B. Ackermann Reduktion):

   $$A = B \wedge B \neq C \wedge a = b \rightarrow A = B \wedge a = c \rightarrow A = C \wedge b = c \rightarrow B = C$$

2. Übersetze Formeln in Aussagenlogik
   - Small Domain Encoding
     - Wenn es n verschiedene Konstanten gibt, gibt es ein Modell mit Größe $\leqslant n$
     - Benutze $\log n$ Bits um den Wert jeder Konstante zu kodieren
     - $a = b$ wird mit den Bits für a und b übersetzt
   - Per-Constraint Encoding
     - Jedes Atom $a = b$ wird mit Variable $P_{a,b}$ ersetzt
     - Transitivitätsconstraints: z.B. $P_{a,b} \wedge P_{b,c} \rightarrow P_{a,c}$

3. Löse entstehende Formel mit SAT Solver

Lazy Encodings

Was sollte ein Theorie Solver können?

Model Generation

Wird der $\mathcal{T}$-Solver auf eine $\mathcal{T}$-konsistente Menge Γ angewandt, kann er ein $\mathcal{T}$-Modell $\mathcal{M}$ mit $\mathcal{M} \models_{\mathcal{T}} \Gamma$ zurückgeben.

Conflict Set Generation

Wird der $\mathcal{T}$-Solver auf eine $\mathcal{T}$-inkonsistente Menge Γ angewandt, kann er eine Teilmenge η von Γ zurückgeben, die den Widerspruch verursacht hat.

Incrementality

Der $\mathcal{T}$-Solver „erinnert“ sich an alte Berechnungen und vermeidet so überflüssigen Rechenaufwand.

Lazy Encodings

Was sollte ein Theorie Solver können?

Backtrackability

Der $\mathcal{T}$-Solver kann Berechnungsschritte effizient rückgängig machen und zu einem früheren Zustand zurückkehren.

Deduction of Unassigned Literals

Auf eine $\mathcal{T}$-konsistente Menge Γ angewandt, kann ein $\mathcal{T}$-Solver Deduktionen der Form $\eta \models_{\mathcal{T}} l$ mit $\eta' \subseteq \Gamma$ und l ein unbelegtes Literal liefern.

Deduction of Interface Equalities

Wenn der $\mathcal{T}$-Solver SAT zurückgibt, kann er Deduktionen der Form $\Gamma \models_{\mathcal{T}} e$ vollziehen

- e ist eine Gleichung zwischen Variablen oder Termen, die in Atomen von Γ vorkommen

Offline Integration

Offline Integration

- Einfachste Integrationsform
- DPLL Solver wird als Black Box verwendet

- Eingabeformel φ mit aussagenlogischer Abstraktion φ^p
- Entscheide φ^p mit SAT Solver
- UNSAT: Auch φ ist UNSAT
- SAT mit erfüllender Belegung Γ^p
  - Menge an Literalen Γ, die Γ^p entspricht wird mit $\mathcal{T}$-Solver entschieden
  - Γ ist $\mathcal{T}$-konsistent: Auch φ ist $\mathcal{T}$-konsistent
  - Γ ist $\mathcal{T}$-inkonsistent: $\neg\Gamma^p$ als Klausel zu φ^p hinzufügen und SAT Solver komplett neu starten

Nachteile

- SAT Solver wird jedes mal komplett neu gestartet
- $\mathcal{T}$-Solver bekommt immer nur vollständige Modelle zum entscheiden

Online Integration: DPLL($\mathcal{T}$)

Algorithmus: dpllt(φ, Γ)

Eingabe: $\mathcal{T}$-Formel φ und $\mathcal{T}$-Belegung Γ
Ausgabe: SAT, wenn φ erfüllbar ist, ansonsten UNSAT

---

if $\mathcal{T}$−preprocess$(\varphi, \Gamma) =$ *Conflict* **then return** UNSAT
while *true* **do**
  $\mathcal{T}$−decideNextBranch(φ^p, Γ^p)
  while *true* **do**
    status = $\mathcal{T}$−deduce(φ^p, Γ^p)
    if *status* $=$ $\mathcal{T}$-SAT **then**
      $\Gamma = \mathcal{B}2\mathcal{T}(\Gamma^p)$
      return SAT
    else if *status* $=$ $\mathcal{T}$-CONFLICT **then**
      blevel = $\mathcal{T}$−analyzeConflict(φ^p, Γ^p)
      if blevel $= -1$ **then return** UNSAT
      $\mathcal{T}$−backtrack$(blevel, \varphi^p, \Gamma^p)$
    else
      break

dpllt Algorithmus—Erklärungen 1

$\mathcal{T}$—preprocess

Simplifiziert φ und updated Γ, so dass $\mathcal{T}$-Erfüllbarkeit von $\varphi \wedge \Gamma$ erhalten bleibt (AL Simplifikation + $\mathcal{T}$-Rewriting)

$\mathcal{T}$—decideNextBranch

Wählt nächste Variable aus

$\mathcal{T}$—analyzeConflict

Erweiterung der klassischen DPLL Konfliktanalyse

- Boolescher Konflikt: Boolesche Konfliktmenge η^p und entsprechendes Level
- $\mathcal{T}$-Konflikt: Benutze Konfliktmenge η des $\mathcal{T}$-Solvers und deren AL-Abstraktion η^p

dpllt Algorithmus—Erklärungen 2

$\mathcal{T}$–deduce(φ^p, Γ^p)

Folgert iterativ Boolesche Literale l^p, die durch die aktuelle Belegung impliziert werden (d.h. $\varphi^p \wedge \Gamma^p \models_p l^p$), bis eine der folgenden Bedingungen wahr wird:

1. Γ^p verletzt φ aussagenlogisch, d.h. $\Gamma^p \wedge \varphi^p \models_p \bot$
   - Verhalten wie DPLL
   - Rückgabe: $\mathcal{T}$-CONFLICT
2. Γ^p erfüllt φ^p aussagenlogisch, d.h. $\Gamma^p \models_p \varphi^p$
   - $\mathcal{T}$-Solver wird auf Γ angewandt
   - Wenn $\mathcal{T}$-konsistent, Rückgabe: $\mathcal{T}$-SAT
   - Andernfalls Rückgabe: $\mathcal{T}$-CONFLICT
3. Keine weiteren Literale können mehr gefolgert werden
   - Rückgabewert: $\mathcal{T}$-UNKNOWN
   - Oder: $\mathcal{T}$-Solver wird auf Γ aufgerufen, wenn $\mathcal{T}$-inkonsistent, dann Rückgabe $\mathcal{T}$-CONFLICT (Early Pruning)

dpllt Algorithmus—Erklärungen 3

$\mathcal{T}$–backtrack

Wie Backtracking im DPLL

- $\neg\eta^p$ wird zu φ^p hinzugefügt
- Backtracking zu Level blevel

Erweiterung von DPLL

1. **Deduktion** nicht nur Boolesch ($\Gamma^p \wedge \varphi^p \models_p l^p$) sondern auch in der Theorie ($\Gamma \models_\mathcal{T} l$)
2. Nicht nur Boolesche **Konflikte** ($\varphi^p \wedge \Gamma^p \models_p \bot$) sondern auch Theorie Konflikte ($\Gamma \models_\mathcal{T} \bot$)

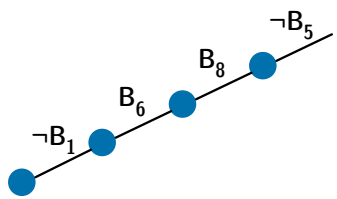
$\mathcal{T}$-DPLL — Beispiel

$\varphi =$		$\varphi^p =$
c_1:	$\{\neg(2x_2 - x_3 > 2) \vee A_1\}$	$\{\neg B_1 \vee A_1\}$
c_2:	$\{\neg A_2 \vee (x_1 - x_5 \leqslant 1)\}$	$\{\neg A_2 \vee B_2\}$
c_3:	$\{(3x_1 - 2x_2 \leqslant 3) \vee A_2\}$	$\{B_3 \vee A_2\}$
c_4:	$\{\neg(2x_3 + x_4 \geqslant 5) \vee \neg(3x_1 - x_3 \leqslant 6) \vee \neg A_1\}$	$\{\neg B_4 \vee \neg B_5 \vee \neg A_1\}$
c_5:	$\{A_1 \vee (3x_1 - 2x_2 \leqslant 3)\}$	$\{A_1 \vee B_3\}$
c_6:	$\{(x_2 - x_4 \leqslant 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1\}$	$\{B_6 \vee B_7 \vee \neg A_1\}$
c_7:	$\{A_1 \vee (x_3 = 3x_5 + 4) \vee A_2\}$	$\{A_1 \vee B_8 \vee A_2\}$

$\mathcal{T}$-DPLL — Beispiel

	$\varphi =$	$\varphi^p =$
c_1:	$\{\neg(2x_2 - x_3 > 2) \vee A_1\}$	$\{\neg B_1 \vee A_1\}$
c_2:	$\{\neg A_2 \vee (x_1 - x_5 \leqslant 1)\}$	$\{\neg A_2 \vee B_2\}$
c_3:	$\{(3x_1 - 2x_2 \leqslant 3) \vee A_2\}$	$\{B_3 \vee A_2\}$
c_4:	$\{\neg(2x_3 + x_4 \geqslant 5) \vee \neg(3x_1 - x_3 \leqslant 6) \vee \neg A_1\}$	$\{\neg B_4 \vee \neg B_5 \vee \neg A_1\}$
c_5:	$\{A_1 \vee (3x_1 - 2x_2 \leqslant 3)\}$	$\{A_1 \vee B_3\}$
c_6:	$\{(x_2 - x_4 \leqslant 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1\}$	$\{B_6 \vee B_7 \vee \neg A_1\}$
c_7:	$\{A_1 \vee (x_3 = 3x_5 + 4) \vee A_2\}$	$\{A_1 \vee B_8 \vee A_2\}$

- Initiale Belegung: $\Gamma^p = \{\neg B_5, B_8, B_6, \neg B_1\}$
- Damit erfüllt: c_1, c_4, c_6, c_7
- Keine weitere Propagation mehr möglich
- Erweiterter Fall 3) von $\mathcal{T}-$deduce: $\mathcal{T}$-Solver wird aufgerufen



$\mathcal{T}$-DPLL — Beispiel

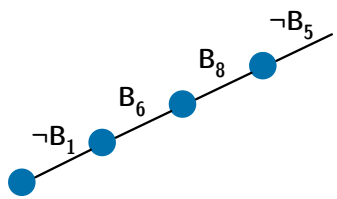
	$\varphi =$	$\varphi^p =$
c_1:	$\{\neg(2x_2 - x_3 > 2) \vee A_1\}$	$\{\neg B_1 \vee A_1\}$
c_2:	$\{\neg A_2 \vee (x_1 - x_5 \leqslant 1)\}$	$\{\neg A_2 \vee B_2\}$
c_3:	$\{(3x_1 - 2x_2 \leqslant 3) \vee A_2\}$	$\{B_3 \vee A_2\}$
c_4:	$\{\neg(2x_3 + x_4 \geqslant 5) \vee \neg(3x_1 - x_3 \leqslant 6) \vee \neg A_1\}$	$\{\neg B_4 \vee \neg B_5 \vee \neg A_1\}$
c_5:	$\{A_1 \vee (3x_1 - 2x_2 \leqslant 3)\}$	$\{A_1 \vee B_3\}$
c_6:	$\{(x_2 - x_4 \leqslant 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1\}$	$\{B_6 \vee B_7 \vee \neg A_1\}$
c_7:	$\{A_1 \vee (x_3 = 3x_5 + 4) \vee A_2\}$	$\{A_1 \vee B_8 \vee A_2\}$

- $\Gamma^p = \{\neg B_5, B_8, B_6, \neg B_1\}$
- $\mathcal{T}$-Solver wird auf

$$\Gamma = \{\neg(3x_1 - x_3 \leqslant 6), (x_3 = 3x_5 + 4),$$

$$(x_2 - x_4 \leqslant 6), \{\neg(2x_2 - x_3 > 2)\}$$

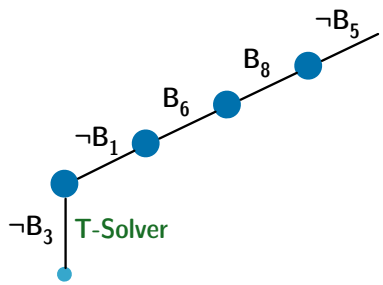
angewendet



$\mathcal{T}$-DPLL — Beispiel

	$\varphi =$	$\varphi^p =$
c_1:	$\{\neg(2x_2 - x_3 > 2) \vee A_1\}$	$\{\neg B_1 \vee A_1\}$
c_2:	$\{\neg A_2 \vee (x_1 - x_5 \leqslant 1)\}$	$\{\neg A_2 \vee B_2\}$
c_3:	$\{(3x_1 - 2x_2 \leqslant 3) \vee A_2\}$	$\{B_3 \vee A_2\}$
c_4:	$\{\neg(2x_3 + x_4 \geqslant 5) \vee \neg(3x_1 - x_3 \leqslant 6) \vee \neg A_1\}$	$\{\neg B_4 \vee \neg B_5 \vee \neg A_1\}$
c_5:	$\{A_1 \vee (3x_1 - 2x_2 \leqslant 3)\}$	$\{A_1 \vee B_3\}$
c_6:	$\{(x_2 - x_4 \leqslant 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1\}$	$\{B_6 \vee B_7 \vee \neg A_1\}$
c_7:	$\{A_1 \vee (x_3 = 3x_5 + 4) \vee A_2\}$	$\{A_1 \vee B_8 \vee A_2\}$

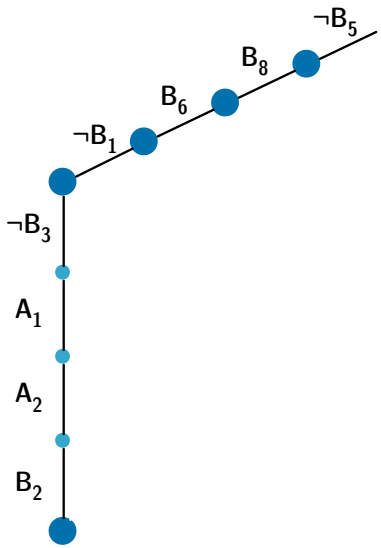
- $\mathcal{T}$-Solver folgert (z.B.) $\neg(3x_1 - 2x_2 \leqslant 3)$ (als Konsequenz des ersten und letzten Literals)
- Entspricht $\neg B_3$
- $\Gamma^p = \{\neg B_5, B_8, B_6, \neg B_1, \neg B_3\}$



$\mathcal{T}$-DPLL — Beispiel

	$\varphi =$	$\varphi^p =$
c_1:	$\{\neg(2x_2 - x_3 > 2) \vee A_1\}$	$\{\neg B_1 \vee A_1\}$
c_2:	$\{\neg A_2 \vee (x_1 - x_5 \leqslant 1)\}$	$\{\neg A_2 \vee B_2\}$
c_3:	$\{(3x_1 - 2x_2 \leqslant 3) \vee A_2\}$	$\{B_3 \vee A_2\}$
c_4:	$\{\neg(2x_3 + x_4 \geqslant 5) \vee \neg(3x_1 - x_3 \leqslant 6) \vee \neg A_1\}$	$\{\neg B_4 \vee \neg B_5 \vee \neg A_1\}$
c_5:	$\{A_1 \vee (3x_1 - 2x_2 \leqslant 3)\}$	$\{A_1 \vee B_3\}$
c_6:	$\{(x_2 - x_4 \leqslant 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1\}$	$\{B_6 \vee B_7 \vee \neg A_1\}$
c_7:	$\{A_1 \vee (x_3 = 3x_5 + 4) \vee A_2\}$	$\{A_1 \vee B_8 \vee A_2\}$

- Unit Propagations: A_1 wegen c_5, A_2 wegen c_3, B_2 wegen c_2
- Dadurch $\Gamma'^p = \{\neg B_5, B_8, B_6, \neg B_1, \neg B_3, A_1, A_2, B_2\}$
- Schicke entsprechendes Γ' an $\mathcal{T}$-Solver
- Rückgabe: UNSAT
- Rückgabe von $\mathcal{T}$–deduce: $\mathcal{T}$-CONFLICT



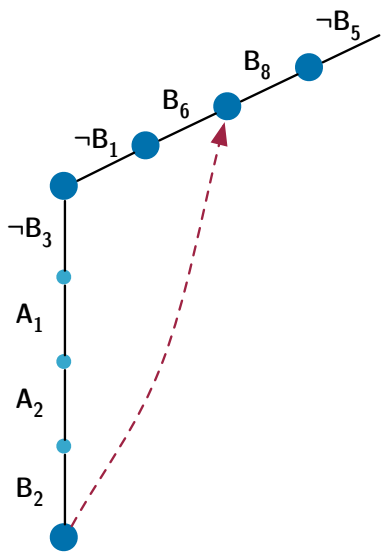
$\mathcal{T}$-DPLL — Beispiel

	$\varphi =$	$\varphi^p =$
c_1:	$\{\neg(2x_2 - x_3 > 2) \vee A_1\}$	$\{\neg B_1 \vee A_1\}$
c_2:	$\{\neg A_2 \vee (x_1 - x_5 \leqslant 1)\}$	$\{\neg A_2 \vee B_2\}$
c_3:	$\{(3x_1 - 2x_2 \leqslant 3) \vee A_2\}$	$\{B_3 \vee A_2\}$
c_4:	$\{\neg(2x_3 + x_4 \geqslant 5) \vee \neg(3x_1 - x_3 \leqslant 6) \vee \neg A_1\}$	$\{\neg B_4 \vee \neg B_5 \vee \neg A_1\}$
c_5:	$\{A_1 \vee (3x_1 - 2x_2 \leqslant 3)\}$	$\{A_1 \vee B_3\}$
c_6:	$\{(x_2 - x_4 \leqslant 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1\}$	$\{B_6 \vee B_7 \vee \neg A_1\}$
c_7:	$\{A_1 \vee (x_3 = 3x_5 + 4) \vee A_2\}$	$\{A_1 \vee B_8 \vee A_2\}$
c_8:	$\ldots$	$\{B_5 \vee \neg B_8 \vee \neg B_2\}$

- $\mathcal{T}$−`analyzeConflict` und $\mathcal{T}$−`backtrack` folgern und lernen die Klausel

$$c_8 = B_5 \vee \neg B_8 \vee \neg B_2$$

- Rücksprung zu entsprechendem blevel
- c_8 ist danach unit



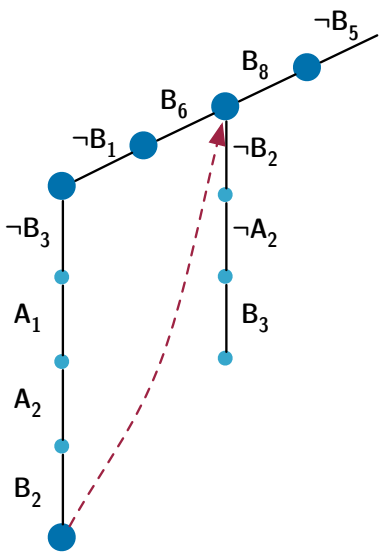
$\mathcal{T}$-DPLL — Beispiel

	$\varphi =$	$\varphi^p =$
c_1:	$\{\neg(2x_2 - x_3 > 2) \vee A_1\}$	$\{\neg B_1 \vee A_1\}$
c_2:	$\{\neg A_2 \vee (x_1 - x_5 \leqslant 1)\}$	$\{\neg A_2 \vee B_2\}$
c_3:	$\{(3x_1 - 2x_2 \leqslant 3) \vee A_2\}$	$\{B_3 \vee A_2\}$
c_4:	$\{\neg(2x_3 + x_4 \geqslant 5) \vee \neg(3x_1 - x_3 \leqslant 6) \vee \neg A_1\}$	$\{\neg B_4 \vee \neg B_5 \vee \neg A_1\}$
c_5:	$\{A_1 \vee (3x_1 - 2x_2 \leqslant 3)\}$	$\{A_1 \vee B_3\}$
c_6:	$\{(x_2 - x_4 \leqslant 6) \vee (x_5 = 5 - 3x_4) \vee \neg A_1\}$	$\{B_6 \vee B_7 \vee \neg A_1\}$
c_7:	$\{A_1 \vee (x_3 = 3x_5 + 4) \vee A_2\}$	$\{A_1 \vee B_8 \vee A_2\}$
c_8:	$\ldots$	$\{B_5 \vee \neg B_8 \vee \neg B_2\}$

- Unit Propagations
  - $\neg B_2$ wegen c_8
  - $\neg A_2$ wegen c_2
  - B_3 wegen c_3
- Alle Klauseln sind erfüllt

$\Rightarrow$ $\mathcal{T}$−deduce gibt $\mathcal{T}$-SAT zurück

$\Rightarrow$ $\mathcal{T}$-DPLL gibt SAT zurück



Kombination von Theorien

Beispiel

Häufig müssen verschiedene Theorien kombiniert werden:

- Lineare Arithmetik und Uninterpretierte Funktionen:

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

- Bit Vektoren und Uninterpretierte Funktionen:

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

- Arrays und lineare Arithmetik

$$x = a\{i \leftarrow e\}[j] \wedge y = a[j] \wedge x > e \wedge x > y$$

Idee von Nelson-Oppen Methode

- Eigenen Solver für jede Theorie
- Solver können Informationen zwischen einander austauschen

Konvexe Theorien

Definition (Konvexe Theorie)

Eine Σ-Theorie $\mathcal{T}$ ist konvex, wenn für jede konjunktive Σ-Formel φ gilt:

$$(\varphi \Rightarrow \bigvee_{i=1}^{n} x_i = y_i) \text{ ist } \mathcal{T}\text{-valide für ein endliches } n > 1 \Longrightarrow$$
$$(\varphi \Rightarrow x_i = y_i) \text{ ist } \mathcal{T}\text{-valide für ein } i \in \{1, \ldots, n\}$$

mit x_i, y_i Variablen.

D.h. Wenn eine Formel eine Disjunktion von Gleichungen impliziert, impliziert sie mindestens eine dieser Gleichungen separat.

Beispiel (Konvexe Theorien)

- Lineare Arithmetik über $\mathbb{R}$ ist konvex
- Lineare Arithmetik über $\mathbb{Z}$ ist nicht konvex
  - $x_1 = 1 \wedge x_2 = 2 \wedge 1 \leqslant x_3 \wedge x_3 \leqslant 2 \Rightarrow (x_3 = x_1 \vee x_3 = x_2)$ gilt
  - $x_1 = 1 \wedge x_2 = 2 \wedge 1 \leqslant x_3 \wedge x_3 \leqslant 2 \Rightarrow x_3 = x_1$ gilt nicht
  - $x_1 = 1 \wedge x_2 = 2 \wedge 1 \leqslant x_3 \wedge x_3 \leqslant 2 \Rightarrow x_3 = x_2$ gilt nicht

Nelson-Oppen Methode—Voraussetzungen

Kombination von Theorien

Im Allgemeinen ist die Kombination von Theorien (selbst von entscheidbaren Theorien) nicht entscheidbar.

Um die Nelson-Oppen Methode anwenden zu können, müssen die Theorien $\mathcal{T}_1, \ldots, \mathcal{T}_n$ folgende Eigenschaften erfüllen:

1. $\mathcal{T}_1, \ldots, \mathcal{T}_n$ sind quantorenfreie First-Order Theorien mit Gleichheit
2. Es gibt eine Entscheidungsprozedur für $\mathcal{T}_1, \ldots, \mathcal{T}_n$
3. Die Signaturen sind disjunkt, d.h. für alle $1 \leqslant i < j \leqslant n, \Sigma_i \cup \Sigma_j = \emptyset$
4. $\mathcal{T}_1, \ldots, \mathcal{T}_n$ werden über unendlichen Domänen interpretiert (z.B. lineare Arithmetik über $\mathbb{R}$, aber nicht Theorie der endlich breiten Bit Vektoren)

Es gibt Erweiterungen von Nelson-Oppen für jede dieser Restriktionen

Nelson-Oppen für konvexe Theorien

Algorithmus: `nelsonOppenConvex`(φ)

Eingabe: eine Konjunktion φ in verschiedenen konvexen Theorien
Ausgabe: `SAT`, wenn φ erfüllbar ist, `UNSAT` sonst

---

1. **Purification:** Purifizieren von φ in $F_1, \ldots, F_n$
2. Wende Entscheidungsverfahren für $\mathcal{T}_i$ auf F_i an
   - Wenn ein i existiert, so dass F_i in $\mathcal{T}_i$ nicht erfüllbar ist, Rückgabe: `UNSAT`
3. **Equality Propagation:** Wenn i und j existieren, so dass
   - F_i eine Gleichung zwischen Variablen in φ $\mathcal{T}_i$-impliziert und
   - diese Gleichung nicht von F_j $\mathcal{T}_j$-impliziert wird,

   dann füge diese Gleichung zu F_j hinzu und **gehe zu Schritt 2)**
4. **Rückgabe:** `SAT`

Purification

- Erfüllbarkeitsäquivalente Transformation einer Formel φ zu φ'
- In φ' ist jede atomare Formel aus nur einer Theorie (*ist pur*)

Vorgehen:

1. $\varphi' := \varphi$
2. Für jeden „fremden“ Teilausdruck ψ in φ'
   - Ersetze ψ mit neuer Hilfsvariable a_ψ
   - Füge Constraint $a_\psi = \psi$ zu φ'

Beispiel

Lineare Arithmetik + Uninterpretierte Funktionen:

$$\varphi = x_1 \leqslant f(x_1)$$

Nach Purifikation:

$$\varphi' = x_1 \leqslant a \wedge a = f(x_1)$$

Alle Atome sind nun pur.

Equality Propagation—1

1. Für alle i: F_i gehört zu $\mathcal{T}_i$ und ist eine Konjunktion von $\mathcal{T}_i$-Literalen
2. Geteilte (*shared*) Variablen sind erlaubt
3. φ ist in der kombinierten Theorie erfüllbar, gdw. $\bigwedge_{i=1}^{n} F_I$ in der kombinierten Theorie erfüllbar ist

Beispiel (Purifikation)

$(f(x_1, 0) \geqslant x_3) \wedge (f(x_2, 0) \leqslant x_3) \wedge (x_1 \geqslant x_2) \wedge (x_2 \geqslant x_1) \wedge (x_3 - f(x_1, 0) \geqslant 1)$

mischt lineare Arithmetik und Uninterpretierte Funktionen.

Purifikation:

$(a_1 \geqslant x_3) \wedge (a_2 \leqslant x_3) \wedge (x_1 \geqslant x_2) \wedge (x_2 \geqslant x_1) \wedge (x_3 - a_1 \geqslant 1) \wedge$
$(a_0 = 0) \wedge$
$(a_1 = f(x_1, a_0) \wedge$
$(a_2 = f(x_2, a_0))$

Optimierungen:

- Ersetze 2-maliges Vorkommen von 0 mit a_0
- Beide Instanzen von $f(x_1, 0)$ werden auf die selbe Hilfsvariable abgebildet

Equality Propagation—2

Beispiel (Equality Propagation)

$(a_1 \geqslant x_3) \wedge (a_2 \leqslant x_3) \wedge (x_1 \geqslant x_2) \wedge (x_2 \geqslant x_1) \wedge (x_3 - a_1 \geqslant 1) \wedge (a_0 = 0) \wedge (a_1 = f(x_1, a_0) \wedge (a_2 = f(x_2, a_0))$

F_1 (Arithmetik über $\mathbb{R}$)	F_2 (EUF)
$a_1 \geqslant x_3$	$a_1 = f(x_1, a_0)$
$a_2 \leqslant x_3$	$a_2 = f(x_2, a_0)$
$x_1 \geqslant x_2$	
$x_2 \geqslant x_1$	
$x_3 - a_1 \geqslant 1$	
$a_0 = 0$	

Equality Propagation—2

Beispiel (Equality Propagation)

$(a_1 \geqslant x_3) \wedge (a_2 \leqslant x_3) \wedge (x_1 \geqslant x_2) \wedge (x_2 \geqslant x_1) \wedge (x_3 - a_1 \geqslant 1) \wedge (a_0 = 0) \wedge (a_1 = f(x_1, a_0) \wedge (a_2 = f(x_2, a_0))$

F_1 (Arithmetik über $\mathbb{R}$)	F_2 (EUF)
$a_1 \geqslant x_3$ $a_2 \leqslant x_3$ $x_1 \geqslant x_2$ $x_2 \geqslant x_1$ $x_3 - a_1 \geqslant 1$ $a_0 = 0$	$a_1 = f(x_1, a_0)$ $a_2 = f(x_2, a_0)$
$x_1 = x_2$	$x_1 = x_2$

- Aus $(x_1 \geqslant x_2) \wedge (x_2 \geqslant x_1)$ folgere $(x_1 = x_2)$
- Propagiere Gleichung nach F_2

Equality Propagation—2

Beispiel (Equality Propagation)

$(a_1 \geqslant x_3) \wedge (a_2 \leqslant x_3) \wedge (x_1 \geqslant x_2) \wedge (x_2 \geqslant x_1) \wedge (x_3 - a_1 \geqslant 1) \wedge (a_0 = 0) \wedge (a_1 = f(x_1, a_0) \wedge (a_2 = f(x_2, a_0))$

F_1 (Arithmetik über $\mathbb{R}$)	F_2 (EUF)
$a_1 \geqslant x_3$ $a_2 \leqslant x_3$ $x_1 \geqslant x_2$ $x_2 \geqslant x_1$ $x_3 - a_1 \geqslant 1$ $a_0 = 0$	$a_1 = f(x_1, a_0)$ $a_2 = f(x_2, a_0)$
$x_1 = x_2$ $a_1 = a_2$	$x_1 = x_2$ $a_1 = a_2$

- Wegen $x_1 = x_2$ folgere $a_1 = a_2$
- Propagiere Gleichung nach F_1

Equality Propagation—2

Beispiel (Equality Propagation)

$(a_1 \geqslant x_3) \wedge (a_2 \leqslant x_3) \wedge (x_1 \geqslant x_2) \wedge (x_2 \geqslant x_1) \wedge (x_3 - a_1 \geqslant 1) \wedge (a_0 = 0) \wedge (a_1 = f(x_1, a_0) \wedge (a_2 = f(x_2, a_0))$

F_1 (Arithmetik über $\mathbb{R}$)	F_2 (EUF)
$a_1 \geqslant x_3$	$a_1 = f(x_1, a_0)$
$a_2 \leqslant x_3$	$a_2 = f(x_2, a_0)$
$x_1 \geqslant x_2$	
$x_2 \geqslant x_1$	
$x_3 - a_1 \geqslant 1$	
$a_0 = 0$	
$x_1 = x_2$	$x_1 = x_2$
$a_1 = a_2$	$a_1 = a_2$
$a_1 = x_3$	$a_1 = x_3$

- Wegen $a_1 = a_2$ folgere $a_1 = x_3$
- Propagiere Gleichung nach F_2

Equality Propagation—2

Beispiel (Equality Propagation)

$(a_1 \geqslant x_3) \wedge (a_2 \leqslant x_3) \wedge (x_1 \geqslant x_2) \wedge (x_2 \geqslant x_1) \wedge (x_3 - a_1 \geqslant 1) \wedge (a_0 = 0) \wedge (a_1 = f(x_1, a_0) \wedge (a_2 = f(x_2, a_0))$

F_1 (Arithmetik über $\mathbb{R}$)	F_2 (EUF)
$a_1 \geqslant x_3$	$a_1 = f(x_1, a_0)$
$a_2 \leqslant x_3$	$a_2 = f(x_2, a_0)$
$x_1 \geqslant x_2$	
$x_2 \geqslant x_1$	
$x_3 - a_1 \geqslant 1$	
$a_0 = 0$	
$x_1 = x_2$	$x_1 = x_2$
$a_1 = a_2$	$a_1 = a_2$
$a_1 = x_3$	$a_1 = x_3$

- $a_1 = x_3$ ergibt mit $x_3 - a_1 \geqslant 1$ einen Widerspruch
- Gebe UNSAT zurück

Nicht-konvexe Theorien—1

Beispiel (Scheitern bei nicht-konvexen Theorien)

- Formel $(1 \leqslant x) \wedge (x \leqslant 2) \wedge P(x) \wedge \neg P(1) \wedge \neg P(2)$ mit $x \in \mathbb{Z}$
- Purifikation:
  $(1 \leqslant x) \wedge (x \leqslant 2) \wedge P(x) \wedge \neg P(a_1) \wedge \neg P(a_2) \wedge a_1 = 1 \wedge a_2 = 2$

F_1 (Arithmetik über $\mathbb{Z}$)	F_2 (EUF)
$1 \leqslant x$	$P(x)$
$x \leqslant 2$	$\neg P(a_1)$
$a_1 = 1$	$\neg P(a_2)$
$a_2 = 2$	

- Sowohl F_1 als auch F_2 unabhängig voneinander erfüllbar
- Keine neuen Gleichungen werden impliziert

$\Rightarrow$ Rückgabewert: SAT

- Originalformel ist jedoch UNSAT in der kombinierten Theorie
- Lösung: F_1 impliziert $x = 1 \vee x = 2$; Hinzufügen der Disjunktion und Case Split

Nicht-konvexe Theorien—2

Beispiel (Case Split bei nicht-konvexen Theorien)

- Purifikation:
  $(1 \leqslant x) \wedge (x \leqslant 2) \wedge P(x) \wedge \neg P(a_1) \wedge \neg P(a_2) \wedge a_1 = 1 \wedge a_2 = 2$
- F_1 impliziert $x = 1 \vee x = 2 \Rightarrow$ Case Split

F_1 (LA über $\mathbb{Z}$)	F_2 (EUF)
$1 \leqslant x$	$P(x)$
$x \leqslant 2$	$\neg P(a_1)$
$a_1 = 1$	$\neg P(a_2)$
$a_2 = 2$	
$x = 1$	
$x = a_1$	$x = a_1$
	`false`

F_1 (LA über $\mathbb{Z}$)	F_2 (EUF)
$1 \leqslant x$	$P(x)$
$x \leqslant 2$	$\neg P(a_1)$
$a_1 = 1$	$\neg P(a_2)$
$a_2 = 2$	
$x = 2$	
$x = a_2$	$x = a_2$
	`false`

- In beiden Fällen ist die Rückgabe false
- Rückgabe UNSAT

Nelson-Oppen für nicht-konvexe Theorien

Algorithmus: `nelsonOppenNotConvex`(φ)

Eingabe: Eine Formel φ mit verschiedenen nicht-konvexen Theorien
Ausgabe: SAT, falls φ erfüllbar ist, UNSAT sonst

1. **Purification:** Purifizieren von φ in $\varphi' = F_1, \ldots, F_n$
2. Wende Entscheidungsverfahren für T_i auf F_i an
   - Existiert ein i, mit F_i in T_i nicht erfüllbar $\Rightarrow$ Rückgabe UNSAT
3. **Equality Propagation:** Wenn i und j existieren, so dass
   - F_i eine Gleichung zwischen Variablen in φ' T_i-impliziert und
   - diese Gleichung nicht von F_j T_j-impliziert wird,

   dann füge diese Gleichung zu F_j hinzu und **gehe zu Schritt 2)**
4. **Splitting:** Wenn ein i existiert mit
   - $F_1 \Rightarrow (x_1 = y_1 \vee \cdots \vee x_k = y_k)$ und
   - $\forall j \in \{1, \ldots, k\}. F_i \not\Rightarrow x_i = y_i$

   Rufe Nelson-Oppen rekursiv auf: $\varphi' \wedge x_1 = y_1, \ldots, \varphi' \wedge x_k = y_k$ auf.

   Ist eines der Subprobleme SAT, so gebe SAT zurück, ansonsten UNSAT
5. **Rückgabe** SAT

Literaturtipps

Literaturhinweis

- *D. Kroening & O. Strichman.* **Decision Procedures: An Algorithmic Point of View Chapters 10 & 11.** Springer, 2008.
- *C. Barrett, R. Sebastiani, S. A. Seshia & C. Tinelli.* **Chapter 26—Satisfiability Modulo Theories in Handbook of Satisfiability.** IOS Press, 2009.

- *G. Nelson & D. C. Oppen.* **Simplification by Cooperatring Decision Procedures.** ACM Transactions on Programming Languages and Systems 1(2), 1979.

Web Links

- `http://www.smtlib.org/` — Bibliothek an Problemen, Infos zum Eingabeformat, usw.
- `http://research.microsoft.com/en-us/um/redmond/projects/z3/` — SMT Solver von Microsoft
- `http://verify.inf.usi.ch/opensmt` — Open Source SMT Solver