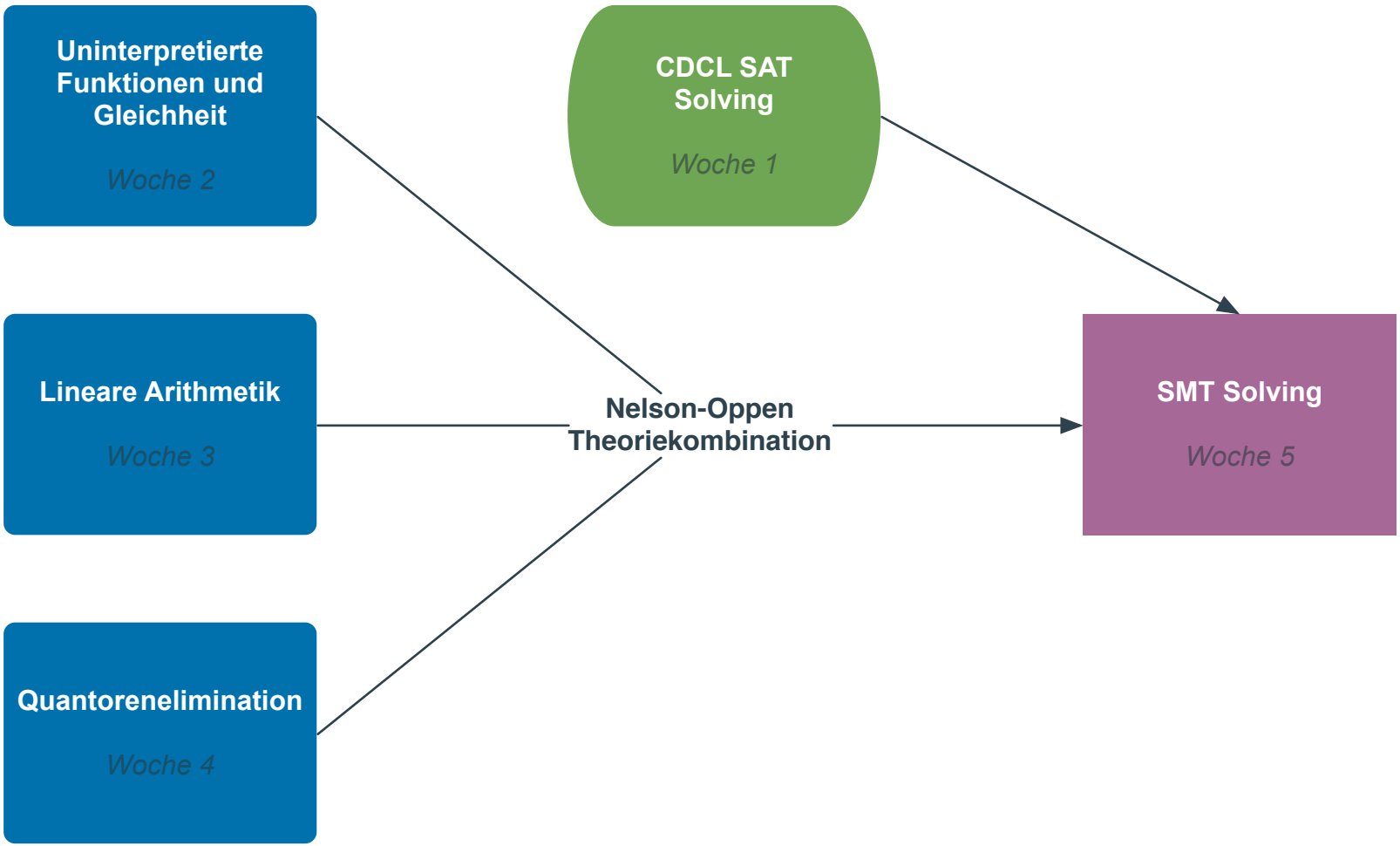
# Automatisches Beweisen—Vertiefung
# Entscheidbare FOL Fragmente
## Lineare Arithmetik

Christoph Zengler

Arbeitsbereich Symbolisches Rechnen
Prof. Dr. Wolfgang Küchlin
Universität Tübingen

6. Dezember 2011 (Nikolaus!)

# Wir erinnern uns...



# Lineare Arithmetik

## Syntax

- **Linear:** Keine Multiplikation zwischen Variablen (d.h. auch keine Potenzen)
- $5x$ ist Abkürzung für $x + x + x + x + x$

## EBNF-Grammatik für Lineare Arithmetik

| | | | |
|---|---|---|---|
| Summe | $=$ | Term \| Summe + Term | |
| Term | $=$ | *Variable* \| *Konstante* \| *Konstante* $\cdot$ *Variable* | |
| Formel | $=$ | $\top \mid \bot$ | Konstanten |
| | $\mid$ | Summe $=$ Summe | |
| | $\mid$ | Summe $\leqslant$ Summe | |
| | $\mid$ | Summe $<$ Summe | |
| | $\mid$ | Formel $\wedge$ Formel | Konjunktion |

- Wir betrachten nur Konjunktionen
- $a - b$ ist *syntactic sugar* für $a + -b$
- $\leqslant$ und $<$ können durch $\geqslant$ und $>$ ersetzt werden (durch Negation der Koeffizienten)

# Lineare Arithmetik

## Semantik

- Universum ist $\mathbb{Q}$ oder $\mathbb{Z}$
- Arithmetische Symbole $+, -, \cdot$ werden wie üblich über $\mathbb{Q}$ und $\mathbb{Z}$ interpretiert
- Konstanten werden als 0-stellige Funktionen mit dem jeweiligen Wert in $\mathbb{Q}$ oder $\mathbb{Z}$ interpretiert
- Prädikate $=, \leqslant, \geqslant$ werden wie üblich über $\mathbb{Q}$ und $\mathbb{Z}$ interpretiert

**Damit ist eine Interpretation $\mathcal{M}_{\mathrm{LA}}$ fixiert.**

- Belegung $\beta$ bildet Variablen nach $\mathbb{Q}$ oder $\mathbb{Z}$ ab
- Im Gegensatz zur allgemeinen FOL reicht für die Erfüllbarkeit die **Existenz** einer Belegung $\beta$ aus, so dass die Formel zu `true` evaluiert.

### Definition (Erfüllbarkeit in der Linearen Arithmetik)

Eine Formel $\varphi$ in LA ist erfüllbar gdw. es eine Belegung $\beta$ gibt, so dass $\texttt{holds}(\mathcal{M}_{\mathrm{LA}}, \beta, \varphi) = \texttt{true}$ gilt.

# $\mathbb{Z}$ vs. $\mathbb{Q}$

Wir betrachten $\mathbb{Z}$ und $\mathbb{Q}$ als Domänen

## ? Was wollen wir wissen?

Ist eine Formel in Linearer Arithmetik erfüllbar, wenn ja mit welcher Belegung der Variablen.

## Beispiel (Formel in Linearer Arithmetik)

$$3x_1 + 2x_2 \leqslant 5x_3 \wedge 2x_1 - 2x_2 = 0$$

- Für $\mathbb{Q}$ ist das Problem in Polynomialzeit entscheidbar
- Für $\mathbb{Z}$ ist das Problem NP-vollständig (Pseudo Boolesche Optimierung)
- Nimmt man in die Sprache noch die “echte” Multiplikation hinzu, so ist das Problem für $\mathbb{Z}$ unentscheidbar (Gödelscher Unvollständigkeitssatz)

# Anwendungsbeispiel

## Code Optimierung im Compiler

### Beispiel (C Programm)

```
for(i=1; i<=10; i++)
  a[j+i]=a[j]
```

- `a[j]` wird 10 mal ausgelesen
- Zugriff auf Speicher ist sehr langsam im Vergleich zu Werten in Registern
- **Frage:** Kann man `a[j]` nur einmal am Anfang der Schleife auslesen?
- **Antwort:** Nur wenn `a[j]` den Wert in der Schleife nicht ändert, also nie auf `a[j]` geschrieben wird

$$i \geqslant 1 \wedge i \leqslant 10 \wedge j + i = j$$

ist nicht erfüllbar, daher ist die Optimierung sicher.

# Übersicht

## Simplex

- Einer der ältesten Algorithmen (Danzig, 1947)
- Finde einen optimalen Wert für eine Zielfunktion bei gegebenen Constraints über reellen Zahlen
- Constraints + Zielfunktion = **Lineares Programm**
- Entscheidungsproblem **Allgemeiner Simplex**: Ist eine gegebene Menge von linearen Constraints über $\mathbb{Q}$ oder $\mathbb{R}$ erfüllbar oder nicht

## Integer Linear Programming (ILP)

- Selbe Fragestellung über $\mathbb{Z}$
- Algorithmus: **Branch and Bound**

## Zwei einfache Ansätze

- **Fourier Motzkin** Variablen Elimination für $\mathbb{R}$ oder $\mathbb{Q}$
- **Omega Test** für $\mathbb{Z}$

# Vorverarbeitung

## Lineare Systeme über $\mathbb{Q}$, $\mathbb{R}$ oder $\mathbb{Z}$

- unabhängig vom gewählten Entscheidungsverfahren

### Erste Vereinfachung

$$x_1 + x_2 \leqslant 2, x_1 \leqslant 1, x_2 \leqslant 1$$

Die erste Ungleichung ist überflüssig

### Allgemeine Formulierung

Für eine Menge

$$S = \left\{ a_0 x_0 + \sum_{j=1}^{n} a_j x_j \leqslant b, l_j \leqslant x_j \leqslant u_j \text{ für } j = 0, \ldots, n \right\}$$

von Constraints ist das erste Constraint redundant, wenn

$$\sum_{j \mid a_j > 0} a_j u_j + \sum_{j \mid a_j < 0} a_j l_j \leqslant b.$$

# Vorverarbeitung

## Lineare Systeme über $\mathbb{Q}$, $\mathbb{R}$ oder $\mathbb{Z}$

### Zweite Vereinfachung

$$2x_1 + x_2 \leqslant 2, x_2 \geqslant 4, x_1 \leqslant 3$$

Aus dem ersten und zweiten Constraint folgt $x \leqslant -1$, d.h. das dritte Constraint kann spezialisiert werden.

### Allgemeine Formulierung

Für $a_0 > 0$ gilt

$$x_0 \leqslant \left( b - \sum_{j \mid j>0, a_j>0} a_j l_j - \sum_{j \mid a_j<0} a_j u_j \right) / a_0$$

und für $a_0 < 0$ gilt

$$x_0 \geqslant \left( b - \sum_{j \mid a_j>0} a_j l_j - \sum_{j \mid j>0, a_j<0} a_j u_j \right) / a_0.$$

# Vorverarbeitung

## Lineare Systeme über $\mathbb{Z}$

Strikte Ungleichungen

$$\sum_{1 \leqslant j \leqslant n} a_i x_i < b$$

können in nicht-strikte Ungleichungen

$$\sum_{1 \leqslant j \leqslant n} a_i x_i \leqslant b - 1$$

umgewandelt werden.

### Hinweis!

Systeme in $\mathbb{Q}$ können durch Multiplikation mit dem kgV der Nenner in Systeme mit rein ganzzahligen Koeffizienten überführt werden.

# Das Simplex Verfahren

- Geht zurück auf Danzig (1947)
- Arbeitet auf nicht-strikten Constraints über $\mathbb{Q}$ (oder $\mathbb{R}$)
- Simplex ist im worst-case exponentiell
- Auf echten Problemen jedoch meistens besser als bekannte polynomiale Algorithmen (z.B. Ellipsoid Methode)

## Eingabe (Allgemeine Form)

Der Allgemeine Simplex akzeptiert zwei verschiedene Arten von Constraints:

1. Gleichheiten der Form

$$a_1 x_1 + \cdots + a_n x_n = 0$$

2. Ober- und Untergrenzen für die Variablen

$$l_i \leqslant x_i \leqslant u_i$$

*Ober und Untergrenzen sind optional.*

# Umwandlung in die Allgemeine Form

Beliebige nicht-strikte lineare Constraints $L \bowtie R$ mit $\bowtie \in \{=, \leqslant, \geqslant\}$ können in die allgemeine Form transformiert werden.

Für jedes Constraint $i$

1. Bewege alle Summanden in $R$ auf die linke Seite, so dass $L' \bowtie b$ mit $b$ Konstante bleibt
2. Führe eine neue Variable $s_i$ ein. Füge die Constraints $L' - s_i = 0$ und $s_i \bowtie b$ hinzu.
   - ist $\bowtie$ gleich $=$, so schreibe $s_i = b$ als $s_i \geqslant b$ und $s_i \leqslant b$

**Benennungen**

- Die $s_i$ werden **Zusatzvariablen** genannt
- $x_i$ sind die **Problemvariablen**

# Umwandlung in die Allgemeine Form

## Beispiel

### Beispiel (Umwandlung in die Allgemeine Form)

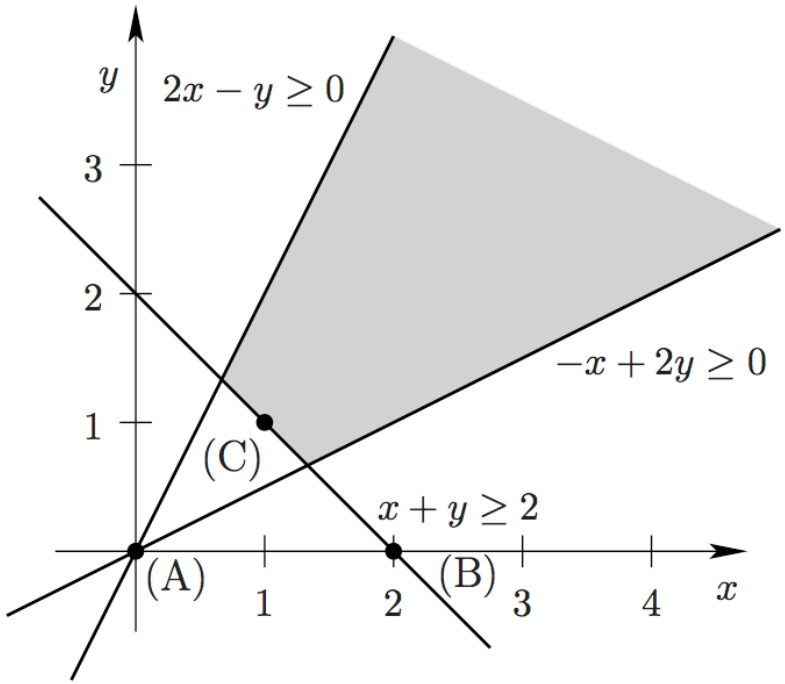
$$\begin{aligned} x + y &\geqslant 2 \wedge \\ 2x - y &\geqslant 0 \wedge \\ -x + 2y &\geqslant 1 \end{aligned}$$

kann in allgemeiner Form geschrieben werden als

$$\begin{aligned} x + y - s_1 &= 0 \wedge \\ 2x - y - s_2 &= 0 \wedge \\ -x + 2y - s_3 &= 0 \wedge \\ s_1 &\geqslant 2 \wedge \\ s_2 &\geqslant 0 \wedge \\ s_3 &\geqslant 1. \end{aligned}$$

# Geometrische Betrachtung

- Jede Variable entspricht einer Dimension
- Ungleichungen definieren Halbräume
- Gleichungen definieren Hyperebenen (Unterraum mit Dimension $n - 1$)




Der abgeschlossene Unterraum der erfüllenden Belegungen ist dann ein Polytop.

# Betrachtung als Matrix

- Koeffizienten des Problems werden als $m \times (n + m)$-Matrix $A$ geschrieben
- Die Variablen $x_1, \ldots, x_n, s_1, \ldots, s_m$ werden als Vektor $\vec{x}$ geschrieben

Erfüllbarkeitsproblem ist dann äquivalent zur Existenz eines Vektors $\vec{x}$, so dass

$$A\vec{x} = 0 \text{ und } \bigwedge_{i=1}^{m} l_i \leqslant s_i \leqslant u_i$$

mit $l_i \in \{-\infty\} \cup \mathbb{Q}$ und $u_i \in \{\infty\} \cup \mathbb{Q}$

## Beispiel (Matrixschreibweise)

$$\begin{aligned} x + y - s_1 &= 0 \wedge \\ 2x - y - s_2 &= 0 \wedge \\ -x + 2y - s_3 &= 0 \wedge \\ s_1 &\geqslant 2 \wedge \\ s_2 &\geqslant 0 \wedge \\ s_3 &\geqslant 1. \end{aligned}$$

$$\begin{pmatrix} 1 & 1 & -1 & 0 & 0 \\ 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & 0 & 0 & -1 \end{pmatrix}$$

mit Variablenreihenfolge

$$x_1, x_2, x_3, s_1, s_2, s_3.$$

# Tableaudarstellung

- Spalten für $s_1, \ldots, s_n$ entsprechen einer $m \times m$ Diagonalmatrix mit Koeffizient $-1$
- $A$ verändert sich im Laufe der Zeit
- Anzahl der Spalten dieser Diagonalmatrix wird niemals reduziert $(= m)$
- Die $m$ Variablen, die zu diesen Spalten gehören heißen **Basisvariablen** $\mathcal{B}$
- **Nicht-Basisvariablen** $\mathcal{N}$

## Das Tableau

- Speichere die Matrix $A$ ohne die Diagonalmatrix
- Tableau ist $m \times n$ Matrix, deren Spalten den Nicht-Basisvariablen entsprechen
- Jede Zeile ist mit einer Basisvariable assoziiert (die -1 in dieser Zeile hat)

# Tableaudarstellung

## Beispiel

### Beispiel (Tableaudarstellung)

$$\begin{pmatrix} 1 & 1 & -1 & 0 & 0 \\ 2 & -1 & 0 & -1 & 0 \\ -1 & 2 & 0 & 0 & -1 \end{pmatrix}$$

als Tableau:

| | $x$ | $y$ |
|---|---|---|
| $s_1$ | 1 | 1 |
| $s_2$ | 2 | $-1$ |
| $s_3$ | $-1$ | 2 |

$A\vec{x} = 0$ kann geschrieben werden als

$$\bigwedge_{x_i \in \mathcal{B}} \left( x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j \right)$$

Als Matrix geschrieben entsprechen die Summen auf der rechten Seite genau dem Tableau.

# Der Simplex Algorithmus

- Zusätzliche Datenstruktur: Belegung der Variablen $\beta : \mathcal{B} \cup \mathcal{N} \to \mathbb{Q}$

## Algorithmus: `generalSimplex`$(S)$

**Eingabe:** Eine Menge $S$ von $m$ linearen nicht-strikten Constraints
**Ausgabe:** `SAT` wenn $S$ erfüllbar ist, `UNSAT` sonst

1. Bringe $S$ in allgemeine Form
2. $\mathcal{B} = \{s_1, \ldots, s_m\}$
3. Konstruiere das Tableau für $A$
4. Belege jede Variable (Basis und Nicht-Basis) mit 0
5. Bestimme eine Ordnung auf den Variablen
6. Gibt es keine Basisvariable, die ihre Grenzen verletzt, gib `SAT` zurück, ansonsten sei $x_i$ die erste Basisvariable in der Ordnung, die ihre Grenzen verletzt
7. Suche die erste Nicht-Basisvariable $x_j$ in der Ordnung, die für die Pivotoperation benutzt werden kann. Gibt es keine solche Variable, gib `UNSAT` zurück
8. Führe die Pivotoperation auf $x_i$ und $x_j$ aus
9. Gehe zu Schritt 6

# Der Simplex Algorithmus

## Die Invarianten

### Invarianten im Algorithmus

INV1 $A\vec{x} = 0$

INV2 Die Werte aller Nicht-Basisvariablen liegen in ihren Grenzen:

$$\text{für alle } x_j \in \mathcal{N} : l_j \leqslant \beta(x_j) \leqslant u_j$$

### Beginn des Algorithmus

- Alle Variablen sind auf 0 gesetzt (INV1) ✓
- Die Nicht-Basisvariablen haben keine Grenzen (INV2) ✓

### Keine Variable verletzt die Grenze

- Wegen (INV1) ist dann $\beta$ eine erfüllende Belegung
- SAT wird zurückgegeben ✓

# Der Simplex Algorithmus

## Die Invarianten

### Eine Basisvariable $x_i$ verletzt ihre Grenzen

- o.B.d.A sei $\beta(x_i) > u_i$, d.h. die Obergrenze wird verletzt

$\Rightarrow$ $x_i$ muss verringert werden

- Definition von $x_i$:

$$x_i = \sum_{x_j \in \mathcal{N}} a_{ij} x_j$$

$\Rightarrow$ $x_i$ kann verringert werden, indem eine Nicht-Basisvariable $x_j$ mit ...

- ... $a_{ij} > 0$ und $\beta(x_j) > l_j$ verringert wird
- ... $a_{ij} < 0$ und $\beta(x_j) < u_j$ erhöht wird

Eine solche Variable heißt passend und kann für die Pivotoperation verwendet werden

- Gibt es keine solche Variable, gib `UNSAT` zurück ✓

# Der Simplex Algorithmus

## Die Invarianten

- $\theta$ ist der Wert, um den wir $\beta(x_j)$ verringern oder erhöhen müssen, um die Obergrenze von $x_i$ zu erfüllen:

$$\theta = \frac{u_i - \beta(x_i)}{a_{ij}}$$

$\Rightarrow$ $x_i$ erfüllt jetzt seine Obergrenze

- $x_j$ kann jedoch nun seine Grenzen verletzten, d.h. (INV2) kann verletzt sein

$\Rightarrow$ $x_i$ und $x_j$ werden im Tableau vertauscht, d.h.

  - $x_i$ wird zur Nicht-Basisvariable
  - $x_j$ wird zur Basisvariable

- Diese Anpassung im Tableau nennen wir Pivotoperation
- Wiederhole die Pivotoperation bis zum Abbruchsfall ✓

# Die Pivotoperation

## Definition (Pivotelement, -spalte und -zeile)

Für zwei Variablen $x_i$ und $x_j$ heißen

- $a_{ij}$ das Pivotelement
- Die Spalte von $x_j$ die Pivotspalte
- Die Zeile $i$ die Pivotzeile

Um $x_i$ und $x_j$ vertauschen zu können muss das Pivotelement $\neq 0$ sein

1. Löse die Zeile $i$ nach $x_j$ auf
2. Eliminiere $x_j$ in allen Zeilen $l \neq i$ durch die Gleichung aus 1)

## Beobachtung

Die Pivotoperation ist ebenfalls die Basisoperation in der Gauss-Elimination

# Beispiel für den Simplex Algorithmus

## Beispiel (Der Simplex Algorithmus)

Initiales Tableau und Grenzen:

| | $x$ | $y$ |
|---|---|---|
| $s_1$ | 1 | 1 |
| $s_2$ | 2 | $-1$ |
| $s_3$ | $-1$ | 2 |

$$2 \leqslant s_1$$
$$0 \leqslant s_2$$
$$1 \leqslant s_3$$

1. $\mathcal{B} = \{s_1, s_2, s_3\}$, $\mathcal{N} = \{x, y\}$
2. Belege alle Variablen mit 0
3. Variablenordnung: $x, y, s_1, s_2, s_3$

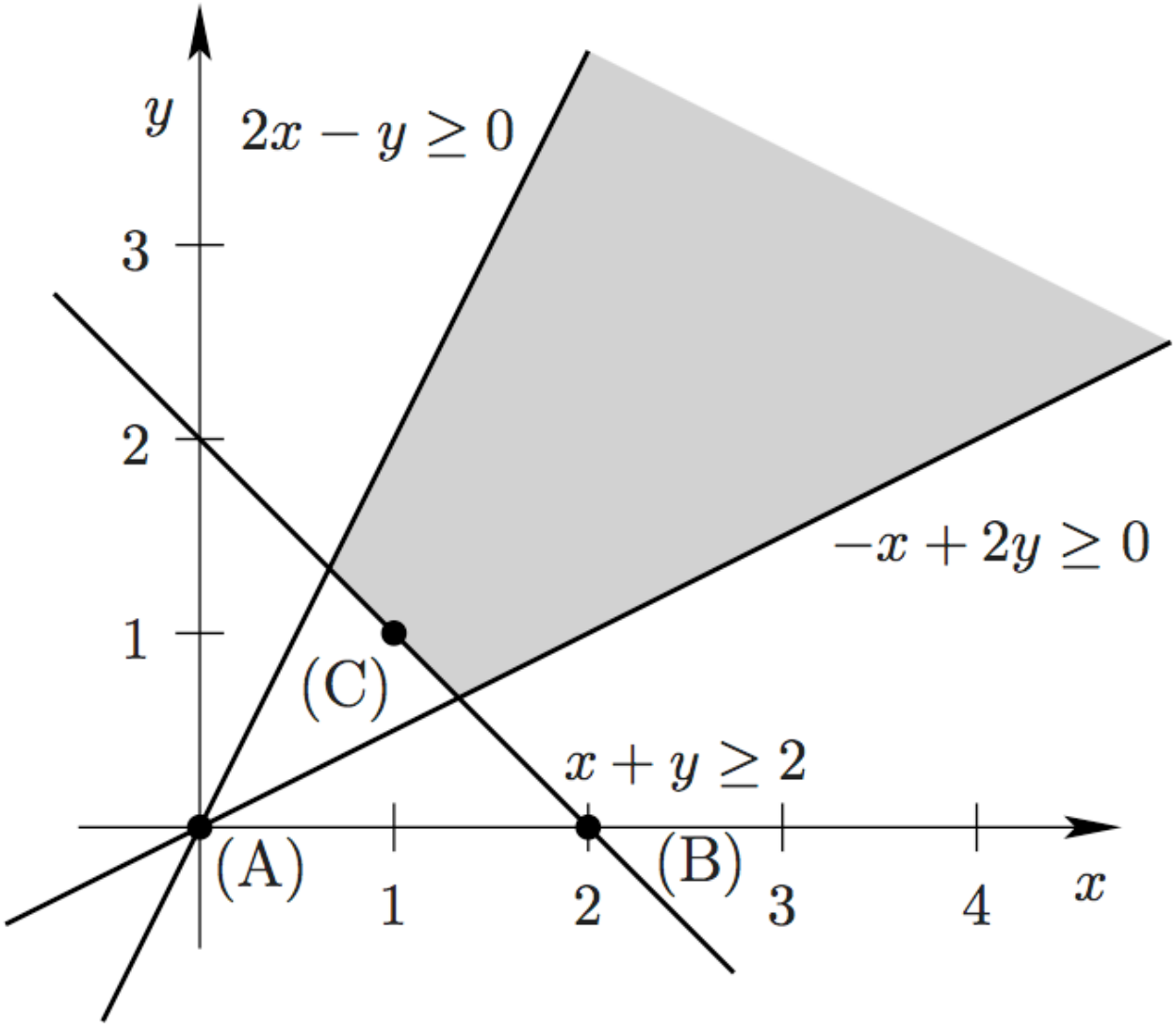
$\Rightarrow$ Erste Basisvariable, die ihre (Unter)grenze verletzt: $s_1 = x + y$

4. Erste Nicht-Basisvariable in der Ordnung $x$ hat positiven Koeffizienten und keine Obergrenze $\Rightarrow$ passend

$\Rightarrow$ $s_1$ muss um 2 erhöht werden, d.h. $x$ muss um 2 erhöht werden

5. Pivotoperation zwischen $s_1$ und $x$

# Beispiel für den Simplex Algorithmus



# Beispiel für den Simplex Algorithmus

## Beispiel (Die Pivotoperation)

| | $x$ | $y$ |
|---|---|---|
| $s_1$ | 1 | 1 |
| $s_2$ | 2 | $-1$ |
| $s_3$ | $-1$ | 2 |

$$\begin{aligned} 2 &\leqslant s_1 \\ 0 &\leqslant s_2 \\ 1 &\leqslant s_3 \end{aligned}$$

### Pivotoperation für $s_1$ und $x$

1. Pivotelement: 1, Pivotspalte: 1, Pivotzeile: 1
2. Löse Zeile 1 nach $x$ auf: $x = s_1 - y$
3. Ersetze $x$ in den anderen beiden Zeilen
   - $s_2 = 2s_1 - 3y$
   - $s_3 = -s_1 + 3y$

### Neues Tableau und Belegungen

| | $s_1$ | $y$ |
|---|---|---|
| $x$ | 1 | $-1$ |
| $s_2$ | 2 | $-3$ |
| $s_3$ | $-1$ | 3 |

$$\beta = \{x \mapsto 2, y \mapsto 0, s_1 \mapsto 2, s_2 \mapsto 4, s_3 \mapsto -2\}$$

# Anmerkungen zum Simplex

## Auswahl des Pivot Elements

- Feste Variablenordnung auf den Basis- und Nicht-Basisvariablen sichert ab, dass keine Menge von Basisvariablen wiederholt wird

$\Rightarrow$ Termination ist garantiert (es können keine Zyklen auftreten)

- Strategie: **Blands Regel**

## Inkrementelle Probleme

- Probleme werden oft inkrementell generiert (siehe Abstraction-Refinement-Loop oder später $\texttt{DPLL}(\mathcal{T})$)
- Allgemeiner Simplex eignet sich gut:
  - Constraints können immer deaktiviert werden indem die Unter- und Obergrenze entfernt werden
  - Pivotoperation auf dem Tableau ist eine Äquivalenzumformung, d.h. erhält die Menge der Lösungen

# Das Branch & Bound Verfahren

- Verbreitetes Verfahren zum Lösen von ganzzahligen linearen Programmen (ILP)
- Betrachtet wie Simplex eigentlich Optimierungsproblem
- Wir betrachten die Entscheidungsvariante
- Eingabe wie bei Simplex: Nicht-strikte lineare Constraints
- Alle Variablen müssen mit Werten aus $\mathbb{Z}$ belegt werden
- Strikte Constraints können durch Vorverarbeitung nicht-strikt gemacht werden

## Definition (Relaxiertes Problem)

Zu einem gegebenen ganzzahligen linearen System $S$ ist das relaxierte Problem $\texttt{relaxed}(S)$ das System $S$ ohne die Bedingung, dass die Variablen aus $\mathbb{Z}$ belegt werden.

**Annahme:**

- Es gibt ein Verfahren $\texttt{LP}_{\text{feasible}}$, dass für ein lineares Programm $S$ entscheidet, ob es erfüllbar ist (dann wird eine Belegung ausgegeben) oder unerfüllbar ist $\Rightarrow$ Variante des Allgemeinen Simplex

# Der Branch & Bound Algorithmus

## Algorithmus: branchAndBound($S$)

**Eingabe:** Ein ganzzahliges lineares System $S$
**Ausgabe:** SAT wenn $S$ erfüllbar ist, UNSAT sonst

---

searchIntegralSolution($S$)
*return* UNSAT

---

**Prozedur** searchIntegralSolution($S$):

$res = \mathsf{LP}_{\text{feasible}}(\texttt{relaxed}(S))$
**if** $res = \text{UNSAT}$ **then**
  *return*
**else**
  **if** $res$ *is integral* **then**
    *abort* SAT
  **else**
    Select a variable $v$ that is assigned a nonintegral value $r$
    searchIntegralSolution($S \cup (v \leqslant \lfloor r \rfloor)$)
    searchIntegralSolution($S \cup (v \geqslant \lceil r \rceil)$)

# Der Branch & Bound Algorithmus

## Illustration an einem Beispiel

### Beispiel (Branch & Bound Algorithmus)

- Variablen von $S$: $x_1, \ldots, x_4$
- Angenommen Lösung von $\mathsf{LP}_{\mathsf{feasible}} = (1, 0.7, 2.5, 3)$

1. Wähle Variable $x_2$
2. Versuche das System $S \cup \{x_2 \leqslant 0\}$ zu lösen
   - Wird keine Lösung gefunden kommt dieser Branch zurück
3. Versuche das System $S \cup \{x_2 \geqslant 1\}$ zu lösen
   - Wird keine Lösung gefunden, gibt es keine Lösung, `searchIntegralSolution` terminiert und `branchAndBound` gibt `UNSAT` zurück
4. Wird eine Lösung gefunden wird das Verfahren iteriert

# Vollständigkeit

## ⚠ Achtung!

Das Verfahren ist so nicht vollständig!

$$1 \leqslant 3x - 3y \leqslant 2$$

hat z.B. keine ganzzahligen Lösungen, jedoch unendlich viele reelle.

## Definition (Small-Model-Property)

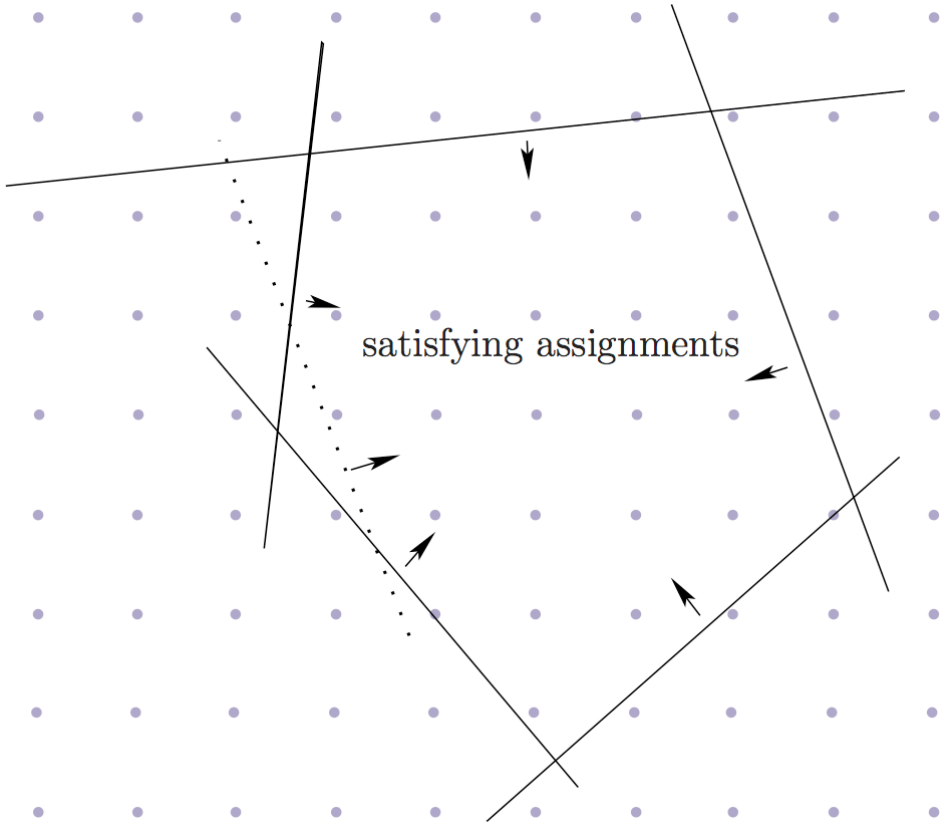
Die Small-Model-Property besagt, dass jede erfüllbare Formel ein Modell mit endlichen Grenzen hat.

- Die Formeln, die wir betrachten, haben diese Small-Model-Property
- Die Grenzen sind berechenbar
- Wir können das Verfahren abbrechen, sobald wir über die berechneten Grenzen einer Variable hinausgehen

# Cutting-Planes

## Idee!

Füge Constraints zu dem Originalsystem hinzu, die den Lösungsraum einschränken, jedoch keine ganzzahligen Lösungen entfernen.



# Cutting-Planes

## Ein Beispiel mit **Gomery Cuts**

- Variablen $x_1, x_2, x_3 \in \mathbb{Z}$
- $1 \leqslant x_1$ und $0.5 \leqslant x_2$
- Das finale Tableau des Allgemeinen Simplex enthält das Constraint

$$x_3 = 0.5x_1 + 2.5x_2$$

und die Belegung $\beta = \{x_1 \mapsto 1, x_2 \mapsto 0.5, x_3 \mapsto 1.75\}$

### Beispiel (Gomery Cut)

Subtrahiere die Belegungen vom Constraint

$$x_3 - 1.75 = 0.5(x_1 - 1) + 2.5(x_2 - 0.5)$$

Umschreiben, so dass die linke Seite ganzzahlig ist:

$$x_3 - 1 = 0.75 + 0.5(x_1 - 1) + 2.5(x_2 - 0.5)$$

- $0.5(x_1 - 1)$ und $2.5(x_2 - 0.5)$ müssen jeweils positiv sein (wg. Untergrenzen von $x_1$ und $x_2$

$\Rightarrow$ Rechte Seite muss ganzzahlig werden, d.h

$$0.75 + 0.5(x_1 - 1) + 2.5(x_2 - 0.5) \geqslant 1$$

$\Rightarrow$ Schließt bisherige Lösung zukünftig aus

# Fourier-Motzkin Variablenelimination

- FM bekommt wie der Allgemeine Simplex eine Konjunktion von linearen Constraints über reellen Variablen
- $m$ Constraints und $x_1, \ldots, x_n$ Variablen

## Auflösen von Gleichungen

Wir betrachten Constraints der Form

$$\sum_{j=1}^{n} a_{ij} \cdot x_j = b_i$$

1. Wähle ein $x_j$ mit Koeffizientem $a_{ij} \neq 0$. O.B.d.A. sei dies $x_n$.
2. Löse die Gleichung nach $x_n$ auf

$$x_n = \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} \cdot x_j$$

3. Substituiere die rechte Seite für $x_n$ im gesamten System $S$ und entferne den Constraint $i$

# Auflösen von Gleichungen

## Beispiel (Auflösen von Gleichungen)

Betrachten wir die Gleichung

$$2x_1 + 3x_2 - 4x_3 = 8$$

Wir lösen die Gleichung nach $x_3$ auf

$$x_3 = 2 - \frac{1}{2}x_1 - \frac{3}{4}x_2$$

und substituieren nun überall die rechte Seite für $x_3$ (d.h. $x_3$ wird aus dem System eliminiert).

Was bleibt ist ein System von $m$ Ungleichungen der Form

$$\bigwedge_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_j \leqslant b_i$$

# Auflösen von Ungleichungen

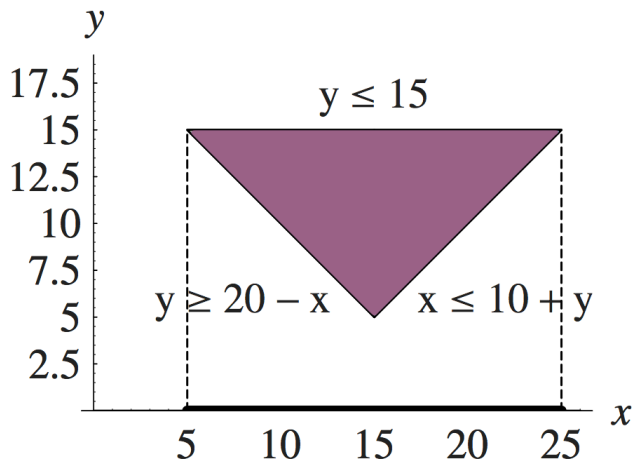
## Idee!

Wähle eine Variable aus und projiziere ihre Constraints auf den Rest des Systems und eliminiere so diese Variable.

## Beispiel (Auflösen einer Ungleichung)

- Die Constraints $x \leqslant y + 10,\ y \leqslant 15,\ y \geqslant -x + 20$ bilden einen Dreieck im $\mathbb{R}^2$
- Projektion auf die $x$-Achse ergibt eine Linie gegeben durch die Constraints

$$5 \leqslant x \leqslant 25$$



# Auflösen von Ungleichungen

- Pro Projektionsschritt wird das Problem um eine Variable kleiner
- Aber möglicherweise mehr Constraints
- Iteriere bis nur noch eine Variable übrig bleibt
- Variablenordnung ist entscheidend (Heuristiken)

---

## Umschreiben der Constraints

Wir eliminieren wieder $x_n$ und betrachten das $i$-te Constraint

$$\sum_{j=1}^{n} a_{ij} x_j \leqslant b_i.$$

- O.B.d.A. betrachten wir den Fall $a_{in} > 0$. Dann kann das Constraint umgeschrieben werden zu

$$a_{in} x_n \leqslant b_i - \sum_{j=1}^{n-1} a_{ij} x_j.$$

# Umschreiben der Constraints

$$a_{in} x_n \leqslant b_i - \sum_{j=1}^{n-1} a_{ij} x_j .$$

- Ist $a_{in} = 0$ braucht das Constraint nicht weiters beachtet werden
- Ansonsten teilen wir durch $a_{in}$:

$$x_n \leqslant \underbrace{\frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} x_j}_{\beta_i} .$$

*(Für $a_{in} > 0$ ist das neue Constraint eine Obergrenze, für $a_{in} < 0$ eine Untergrenze)*

- Die rechte Seite des neuen Constraints bezeichnen wir mit $\beta_i$.

## Idee!

Um eine Variable $x_n$ zu eliminieren, löse alle Ungleichungen, die sie enthalten, nach $x_n$ auf und bilde alle möglichen Paare von Ober- und Untergrenzen auf ihr.

# Eliminieren von Variablen

## Ungebundene Variablen

- Hat eine Variable nach Auflösung in allen Constraints **nur** Unter- oder nur Obergrenzen, können sie und alle Constraints, die sie enthalten einfach entfernt werden.
- Dies kann andere Variablen ungebunden machen, also iteriere diesen Schritt

## Gebundene Variablen

- Zähle alle möglichen Paare aus Untergrenzen $\beta_l$ und Obergrenzen $\beta_u$ auf
- Für jedes Paar gilt dann: $\beta_l \leqslant x_n \leqslant \beta_u$
- Füge das neue Constraint $\beta_l \leqslant \beta_u$ hinzu
- Entsteht dadurch ein Widerspruch, ist das System nicht erfüllbar

# Beispiel für Fourier-Motzkin

$$\begin{array}{rrrrcr} x_1 & -x_2 & & & \leqslant & 0 \\ x_1 & & -x_3 & & \leqslant & 0 \\ -x_1 & +x_2 & +2x_3 & & \leqslant & 0 \\ & & -x_3 & & \leqslant & -1 \\ & & & x_4 & \leqslant & 8 \end{array}$$

## Beispiel (Fourier-Motzkin)

1. $x_4$ ist ungebunden (nur Obergrenze), d.h. $x_4$ und Constraint 5 können entfernt werden
2. Elimination von $x_1$:
   1. Umschreiben der Constraints 1-3:

$$\begin{array}{rcl} x_1 & \leqslant & x_2 \\ x_1 & \leqslant & x_3 \\ x_2 + 2x_3 & \leqslant & x_1 \end{array}$$

   2. Zwei Obergrenzen und eine Untergrenze $\Rightarrow$ Zwei neue Constraints $2x_3 \leqslant 0$ und $x_2 + x_3 \leqslant 0$

# Beispiel für Fourier-Motzkin

Neues Problem nach Elimination von $x_4$ und $x_1$

$$\begin{array}{rrcr} & 2x_3 & \leqslant & 0 \\ x_2 & +x_3 & \leqslant & 0 \\ & -x_3 & \leqslant & -1 \end{array}$$

## Beispiel (Fourier-Motzkin)

1. $x_2$ ist nun ungebunden (nur Obergrenze), d.h. $x_2$ und Constraint 2 können entfernt werden
2. Es bleibt das System

$$\begin{array}{rcr} 2x_3 & \leqslant & 0 \\ -x_3 & \leqslant & -1 \end{array}$$

Kombination des Unter- und Obergrenze von $x_3$ ergibt $1 \leqslant 0$

$\Rightarrow$ Ein Widerspruch, also ist das System nicht erfüllbar

# Anmerkungen zu Fourier-Motzkin

- Im Gegensatz zum Allgemeinen Simplex in seiner präsentierten Form kann FM auch mit strikten Ungleichungen umgehen

---

## Komplexität

- In jedem Schritt kann sich die Anzahl der Constraints im worst-case von $\mathrm{m}$ auf $\mathrm{m}^2/4$ erhöhen
- Insgesamt also $\mathrm{m}^{2^n}/4^n$ Constraints, d.h. doppelt-exponentiell

$\Rightarrow$ Nur für kleine Systeme geeignet

# Literaturtipps

## Literaturhinweis

- *D. Kroening & O. Strichman.* **Decision Procedures: An Algorithmic Point of View Chapter 5.** Springer, 2008.

---

- *G. B. Danzig.* **Linear Programming and Extensions.** Princton University Press, 1963.
- *B. Dutertre & L. de Moura* **Integrating Simplex with DPLL(T).** Technical Report SRI-CSL-06-01, Stanford Research Institute (SRI), 2006.