Rechnergestützte Beweissysteme

Spezielle Entscheidungsverfahren

Entscheidungsverfahren

- Spezielle Gleichungstheorien
  - Rationale und Reelle Zahlen
    - Auflösen und Eliminieren
  - Ganze Zahlen
    - Pugh's Verfahren
  - Uninterpretierte Funktionssymbole
    - Congruence Closure
  - Theorien mit Normalformen
    - Boolesche Algebra
    - Knuth-Bendix
  - Listen (Stacks)
  - Arrays
- Ungleichungen
  - Fourier Motzkin
  - sup-inf-Methode
- Kombination von Theorien
  - Voraussetzungen
  - Nelson-Oppen
  - Beschleunigtes Nelson-Oppen

Theorien

- Sei Σ eine Signatur und $\mathcal{L}(\Sigma)$ die prädikatenlogische Sprache über Σ.
  Wir nehmen an, dass „=„ mit den üblichen Axiomen immer zu Σ gehört.
- Eine *Theorie* T ist eine Menge von Aussagen aus $\mathcal{L}(\Sigma)$, die unter Folgerungen abgeschlossen ist

$$\frac{\Phi \subseteq T \ , \quad \Phi \vdash \phi}{\phi \in \Phi}$$

  - Meist verlangt man noch : false $\notin T$
- Beispiele:
  - $Th(\Sigma_=)$, die Menge aller Tautologien von $\mathcal{L}(\Sigma)$
  - Für jedes Modell $\mathcal{M}=(M,(f_i^M)_{i \in I}, (R_j^M)_{j \in I})$ von Σ ist $Th(\mathcal{M})$ eine Theorie
    - Solche Theorien haben eine besondere Eigenschaft
      - Für jede Aussage aus $\phi \in \mathcal{L}(\Sigma)$ gilt: $\phi \in TH(\mathcal{L})$ oder $\neg \phi \in TH(\mathcal{L})$
    - Also etwa: $Th(\mathbb{Q},0,1,+,-,*,/,=,\leq)$, die Theorie der rationalen Zahlen

Entscheidungsverfahren

- Ein *Entscheidungsverfahren* für eine Theorie T ist ein Algorithmus, der entscheidet, ob eine Aussage ϕ in T ist, oder nicht.
- Eine Theorie T heißt daher *entscheidbar*, falls die charakteristische Funktion χ_{T} berechenbar ist
- Ist $\mathcal{M}$ ein Modell und Th($\mathcal{M}$) die Theorie von $\mathcal{M}$, so gilt: Th($\mathcal{M}$) ist aufzählbar gdw Th($\mathcal{M}$) ist entscheidbar.
- Ziel dieses Kapitels ist ein Überblick über existierende Entscheidungsverfahren praktisch relevanter Theorien.

Triviale Aussagen

- Viele PVS-Beweise enden in einfachen Aussagen über elementare Datentypen

  Beispiel: Der Beweis von

  $\forall n \in nat.\ n \geq 8 \rightarrow \exists x \in nat. \exists y \in nat. 3*x+5*y=n$

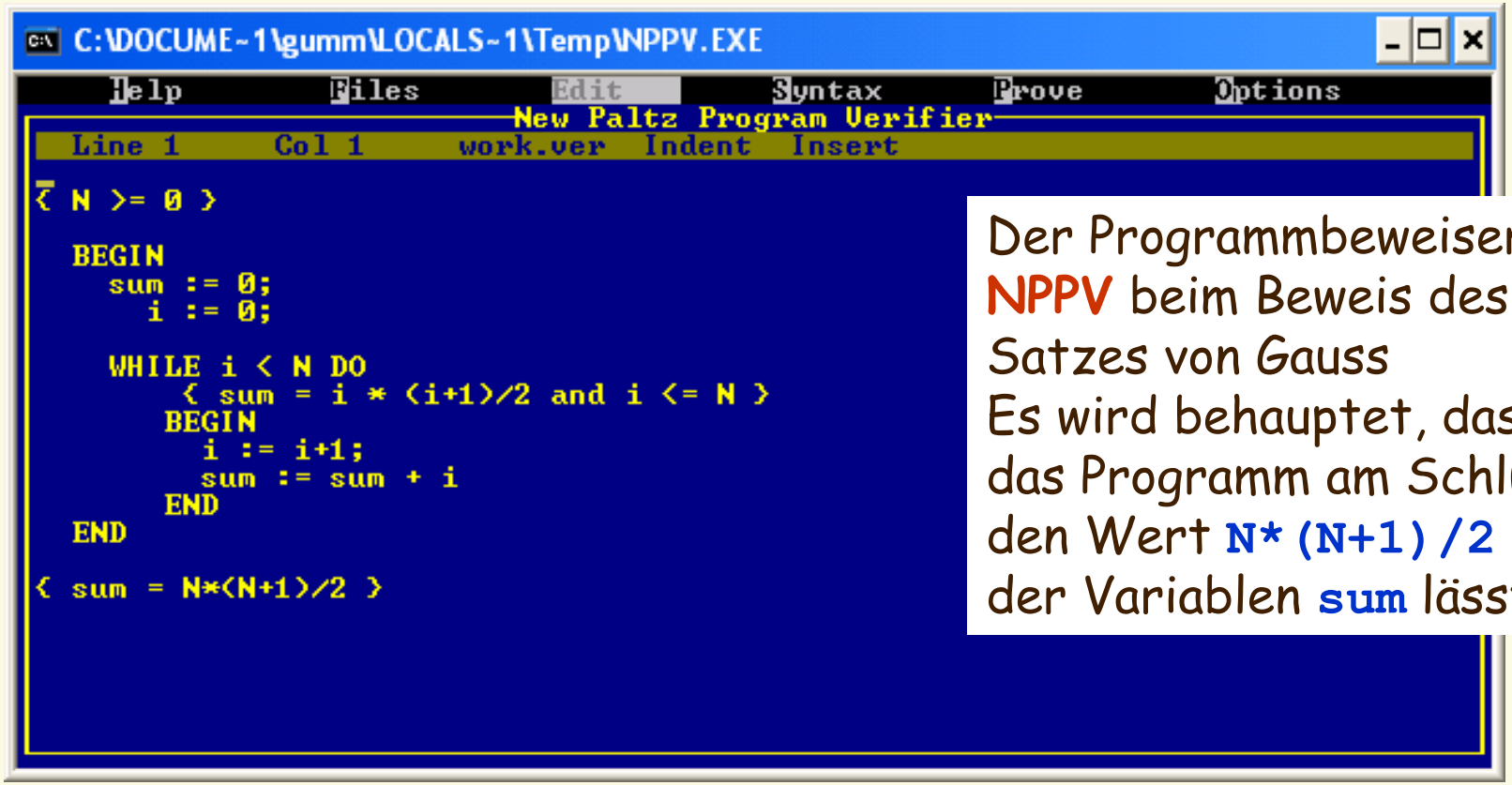
  führt auf die trivialen Behauptungen

  - $0>8 \rightarrow \exists x,y:nat.\ 3*x+5*y=n$
  - $k=8 \rightarrow 3*3+5*0=8+1$
  - $b \geq 1 \wedge 3*a+5*b=k \wedge k>8 \rightarrow 3*(a+2)+5*(b-1)=k+1$
  - $b \geq 1 \wedge 3*a+5*b=k \wedge k>8 \rightarrow b-1 \geq 0$
  - $a>2 \wedge 3*a=k \wedge k>8 \rightarrow 3*(a-3)+5*2=1+k$
  - $3*a=k \wedge k>8 \rightarrow a>2$
  - $3*a+5*b=k \wedge k>8 \rightarrow b=0 \vee b \geq 1$

- Jede davon wird mit (`assert`) erledigt
  - Was passiert dabei ?

Notwendigkeit von Entscheidungsverfahren

- Trivial aussehende Aussagen entstehen auch beim Programmbeweisen



Der Programmbeweiser **NPPV** beim Beweis des Satzes von Gauss
Es wird behauptet, dass das Programm am Schluss den Wert `N*(N+1)/2` in der Variablen `sum` lässt.

Verifikationsbedingungen

- Der Beweis des Programmes führt auf die folgenden 3 Verifikationsbedingungen

  1. $N \geq 0 \rightarrow 0 = 0*(0+1)/2 \ \wedge \ 0 \leq N$
  2. $sum = i*(i+1)/2 \wedge i \leq N \wedge \neg i < N \rightarrow sum = N*(N+1)/2$
  3. $i < N \rightarrow i+1 \leq N$

- Ein Programmverifizierer sollte alle diese und ähnliche einfachen Aussagen automatisch beweisen oder widerlegen
  - also: Entscheidungsverfahren gesucht

- Beachte:
  - die dritte Behauptung gilt für natürliche und für ganze Zahlen,
  - nicht für rationale oder reelle.

Entscheidungsverfahren

- Hier: Gesucht ein Entscheidungsverfahren mit
  - Input: eine (zahlentheoretische) Aussage
  - Output: **true**, falls die Aussage wahr ist,
    false sonst

- Welche Entscheidungsverfahren sind bekannt
  - Wann funktionieren sie
  - Wie funktionieren sie

- Wenn es kein Entscheidungsverfahren gibt
  - sind wir zu dumm, oder
  - ist es prinzipiell unmöglich ?

Verfahren für spezielle Theorien

- Zunächst Entscheidungsverfahren für spezielle Theorien …

1. arithmetische Gleichungen
   - $2*sum=i*(i+1) \wedge i=N \rightarrow sum = N*(N+1)/2$

2. Gleichungen in Datentypen
   - $l \neq null \rightarrow cons(car(l),cdr(l))=l$
   - $a=write(b,k,v) \rightarrow a[i]=(i=k)?v:b[i]$

3. Ungleichungen
   - $b \geq 1 \wedge 3*a+5*b \leq k \wedge k>8 \rightarrow 3*(a+2)+5*(b-1) \leq k+1$

- … später müssen wir die Verfahren kombinieren
  - $i+1 \leq k \wedge a=write(b,k,v) \rightarrow a[i]=b[i]$

Die Arithmetik ist unentscheidbar


Kurt Gödel

- „Arithmetik" ist die Theorie der ganzen Zahlen Th($\mathbb{Z}$,+,*,<, =)
  - Signatur besteht aus:
    - Operationen: 0, 1, +, *
    - Relationen: =, <
  - Die Operationen *div*, *mod* sind *definierbar*:
    - $\mathtt{div(a,b) = c} \leftrightarrow \mathtt{0 \le a-b*c \wedge a-b*c < b}$
  - Beispiel:
    - $\forall n \in Z.\exists x,y \in Z.\ 3*x+5*y=n$ $\in$ Th(Z,+,*,<, =)
    - $\forall x,y \in Z.\exists z \in Z.\ x*x*x+y*y*y=z*z*z$ $\notin$ Th(Z,+,*,<, =)
- Es gibt kein Entscheidungsverfahren für Th(Z,+,*,<,=)
  - Gödel kodiert Turingmaschinen als zahlentheoretische Formeln
  - Das Halteproblem wird zu einer zahlentheoretischen Aussage
- Folgerung: Die Arithmetik ist nicht aufzählbar
- Insbesondere gibt es keine endliche Axiomatisierung (1. Stufe)

Jetzt ein positives Resultat

- Presburger betrachtet zahlentheoretische Formeln ohne Multiplikation
- *Presburger Arithmetik* ist die Theorie der ganzen Zahlen Th($\mathbb{Z}$,0,1,+,-,<, =)
  :
  - Signatur besteht aus:
    - Operationen: 0, 1, +, -
    - Relationen: =, <
  - Es gibt ein Entscheidungsverfahren für die Presburger Arithmetik: „Quantorenelimination"
  - Für eine **Konstante** a $\in \mathbb{Z}$ steht a*x als Abkürzung für $\overbrace{x+x+\ldots+x}^{\text{a-mal}}$
- Leider
  - ist das Verfahren ist hyperexponentiell
  - für praktische Zwecke irrelevant

$$2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$$

Offene Formeln

- Wir beschränken uns auf Aussagen einer bestimmten Bauart:
  - Offene Formeln
    - Formeln ohne Quantoren
    - Menge von Klauseln
    - Gedanklich: alle Variablen allquantifiziert
- Interaktiver Beweiser muss nur alle Quantoren entfernen
  - danach setzt der Entscheider ein
  - man hat eine Aussage ohne Quantoren,
    evtl. mit Skolemkonstanten
  - erfüllbar gdw. die entsprechende offene Formel
    mit Variablen anstelle von Skolemkonstanten erfüllbar

Entscheidung für offene Formeln

- Φ sei offene Formel, die zu zeigende Aussage also $\forall x.\ \Phi$
- Schreibe Φ als Konjunktion von $\vee$-Klauseln κ_i, i=0,…,n-1.

  $$\Phi = \wedge_{i<n}\ \kappa_i$$

  dann gilt $\forall x.\Phi = \forall x.\ \wedge_i \kappa_i = \wedge_i \forall x.\ \kappa_i$
- Es reicht also, $\vee$-Klauseln

  $$p_1 \wedge \ldots \wedge p_k \rightarrow q_1 \vee \ldots \vee q_n$$
- beweisen zu können, bzw. zeigen zu können, dass deren Komplemente, $\wedge$-Klauseln,

  $$p_1 \wedge \ldots \wedge p_k \wedge \neg q_1 \ldots \wedge \neg q_n$$

  nicht erfüllbar sind.
- Im Folgenden beschäftigen wir uns mit der Frage, wie man entscheiden kann, ob eine Konjunktion von Literalen erfüllbar ist.

Solver

- Ein *Solver* für eine Theorie Th ist ein Algorithmus, der jeden atomaren Ausdruck $A(x_1,\ldots,x_n)$ in folgendem Sinne lösen kann:

  1. solve($A(x_1,\ldots,x_n)$) = *false*, gdw. es keine $t_1,\ldots,t_n$ mit $A(t_1,\ldots,t_n)$ gibt.

  2. solve($A(x_1,\ldots,x_n)$) = $\{ x_1=t_1, \ldots, x_n=t_n \}$, ansonsten. Dabei

     - darf keine der Variablen $x_1, \ldots, x_n$ in den Termen $t_1,\ldots,t_n$ vorkommen.
     - Sind $z_1,\ldots,z_k$ die Variablen in $t_1,\ldots,t_n$ so muss gelten:

       $$Th \models \forall x_1,..,x_k.\ (A(x_1,\ldots,x_n) \leftrightarrow \exists\ z_1,..,z_r.\ x_1= t_1 \wedge \ldots \wedge x_n=t_n)$$

- Beispiele:
  - Lineare Arithmetik über $\mathbb{Q}$
    - solve($ax+by=c$) = $\{ x = (b/a)*z-(c/a), y = z \}$
  - Lineare Arithmetik über $\mathbb{Z}$
    - solve basiert auf dem Euklidischen Algorithmus
      - solve($3x+5y=1$) = $\{ x = -3+5z,\ y = 2-3z \}$
      - solve($2x+6y=3$) = false

Lösung der homogenen Gleichungng

$(x,y)=(x_0,y_0)+z(a,-b)$

Partikularlösung

Euklidscher Algorithmus

```
int ggT(int a, int b){
      int u=a, v=b;
      while(u != v){
        if(u>v)
           u-=v;
        else
           v-=u;
      }
      return u;
   }
}
```

$ggT(a,b)=d \Rightarrow \exists\, x,y \in Int.\ d=x*a+y*b$

Invariante: u[0] = u[1]*a+u[2]*b
&& v[0] = v[1]*a+v[2]*b

```
1  public class Euklid{
2
3    void euklid(int a, int b){
4         int[] u={a,1,0}, v={b,0,1};
5         while(u[0] != v[0]){   assert    u[0]==a*u[1]+b*u[2]
6                                       && v[0]==a*v[1]+b*v[2];
7           if(u[0]>v[0]){
8              u[0]-=v[0];u[1]-=v[1];u[2]-=v[2];
9           }else{
10             v[0]-=u[0];v[1]-=u[1];v[2]-=u[2];
11          } }
12        System.out.println(
13         "ggT("+a+","+b+") = "+u[0]+" = " +u[1]+"*"+a+" + "+u[2]+"*" +b);
14
15     }
```

BlueJ: Terminal Window - test

Options

```
Der ggT(23,34) = 1 = 3*23 + -2*34
Der ggT(123,345) = 3 = 101*123 + -36*345
```

Nachbedingung:
u[0] = ggT(a,b) = u[1]*a+u[2]*b

Einsatz eines Solvers

- Existiert ein Solver, so können offene Formeln

$$p_1 \wedge p_2 \wedge \ldots \wedge p_k \wedge \neg p_{k+1} \wedge \ldots \wedge \neg p_n$$

folgendermaßen vereinfacht werden:

  - Falls solve(p_1) = false, ist die Formel nicht erfüllbar, ansonsten

$$\text{solve}(p_1) = \sigma = \{x_1 = t_1, \ldots, x_k = t_k\}$$

  - Entferne p_1 und wende die Substitution auf den Rest der Klausel an: $\sigma(p_2) \wedge \ldots \wedge \sigma(p_k) \wedge \neg\sigma(p_{k+1}) \wedge \ldots \wedge \neg\sigma(p_n)$

  - Zum Schluss hat die Klausel die Form

$$\neg q_{k+1} \wedge \ldots \wedge \neg q_n$$

mit atomaren Ausdrücken $\neg q_{k+1}, \ldots, \neg q_n$.

In vielen Theorien sind solche Formeln immer erfüllbar

Solver für Lineare Gleichungen

- Signatur: ($\mathbb{K}$, +, 0, 1, $(c_i)_{i\in I}$, =) -- z.B. $\mathbb{K} = \mathbb{Q}, \mathbb{R}, \mathbb{F}_q$
  - die c_i sind Konstanten
  - „-" ist definierte Operation: **a-b = c ↔ c+b = a**
- Entscheidungsverfahren für die Theorien
  - TH($\mathbb{Q}$,+,0,1, $(c_i)_{i\in I}$,=) Rationale Zahlen
  - TH($\mathbb{R}$,+,0,1, $(c_i)_{i\in I}$,=) Reelle Zahlen
- Gemeinsame Idee
  - Jeder Term kann auf Normalform gebracht werde
    $t(x_1,\ldots,x_n) = a_0 + a_1x + \ldots + a_nx_n$ mit $a_i \neq 0$
  - Jeder atomare Ausdruck hat dann die Form
    eq : $\Sigma_{ii}\, a_ix_i = a_0$ mit $i \in \{0, 1, \ldots, n\}$.

Lineare Gleichungen über $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{F}_q$

- Wähle eine Gleichung
  - $\Sigma_i a_i x_i = a_0$
  - und eine Variable darin mit Koeffizient $\neq 0$, z.B. x_k
- Löse nach x_k auf
  - $x_k = (1/a_k)\,(\,a_0 - \Sigma_{i \neq k}\, a_i x_i\,)$
- Setze x_k in alle anderen Gleichungen ein
- Zwischenergebnis:
  - Eine Variable weniger
  - Bei einer $\wedge$ -Klausel $eq_1 \wedge \ldots \wedge eq_n \wedge \neg eq_{n+1} \wedge \ldots \wedge \neg eq_{n+k}$ ein positives Literal weniger
- Zum Schluss:
  - Wenn eins der positiven Literale nicht mehr erfüllbar, dann ist die Klausel nicht erfüllbar
  - Wenn nur noch nichttriviale negative Literale übrigbleiben, so ist die Klausel erfüllbar. (Konjunktionen von Formeln der Art $\Sigma_i a_i x_i \neq a_0$ sind immer erfüllbar)

Lineare Gleichungen über $\mathbb{Z}$

- Theorie der Standardinterpretation $(\mathbb{Z},+,0,1,=)$
  - Jeder Term kann in die Form $\Sigma_i a_i x_i$ gebracht werden,
  - jede atomare Aussage in: $\Sigma_i a_i x_i = a_0$
- Verwende Euklidischen Algorithmus
  - Gegeben $\Sigma_i a_i x_i = c$
  - Prüfe, ob $d = ggT(a_1,\ldots,a_n) \mid c$
  - Wenn nein, gibt es keine Lösung
  - Ansonsten: Nach Euklid gibt es $d_1,\ldots,d_n \in \mathbb{Z}$ mit $\Sigma_i a_i d_i = d$
    - damit ist $(c/d)*(d_1,\ldots,d_n)$ eine Partikularlösung
    - sei oB.d.A. $|a_n| \leq |a_i|$ für $i < n$.

    Mit neuen Variablen $z_1,\ldots,z_n$ ist

    $(a_n z_1,\ldots,a_n z_{n-1},-(a_1 z_1+\ldots+a_{n-1} z_{n-1}))$ Lösung der homogenen Gleichung
  - also ist
  - solve($\Sigma_i a_i x_i = d$) :

    $(x_1,\ldots,x_{n-1},x_n) = (c/d)*(d_1,\ldots,d_{n-1},d_n) + (a_n z_1,\ldots,a_n z_{n-1},-(a_1 z_1+\ldots+a_{n-1} z_{n-1}))$

    wobei $z_1,\ldots,z_n \in \mathbb{Z}$

Lineare Ungleichungen

- Auf den arithmetischen Datentypen hat man meist die Relation $\leq$.
  - Betrachte die Theorie $\mathrm{Th}(\mathbb{Z}, (+,-,0,\mathrm{succ}),(\leq,=))$
    - $=$ kann man auch als definierte Relation auffassen
    - $<$ ebenso
  - Atomare Ausdrücke sind von der Form $\Sigma_i\, a_i\, x_i \leq 0$
  - Literale ebenso, denn $\neg(u \leq v) \leftrightarrow u > v \leftrightarrow v+1 \leq u$
- Die Theorie $\mathrm{Th}(\mathbb{Z}, (+,-,0,\mathrm{succ}),(\leq,=))$ ist entscheidbar (Presburger)
  - wir wissen schon
    - zu ineffizient, nicht praktikabel
  - für offene Formeln aber praktikabel:
    - Simplex-Methode
    - Fourier-Motzkin

Fourier-Motzkin

- Gegeben: System von Ungleichungen
- Wähle eine Variable x und multipliziere mit sign(x)
  - jede Ungleichung, in der x vorkommt, liefert entweder
    - untere Schranke von x, oder
    - oder obere Schranke von x
- Hat x nur untere (nur obere) Schranken, so ist das System lösbar
  - beliebige Werte für andere Variablen,
  - x groß genug (klein genug)

- Alle Koeffizienten von x positiv

$$x + y \geq 16 \quad (1)$$
$$4x + 7y \leq 28 \quad (2)$$
$$2x - 7y \leq 20 \quad (3)$$
$$2x - 3y \geq -9 \quad (4)$$

- Für festes y sind (1) und (4) untere Schranken für x
- Für festes y sind (2) und (3) obere Schranken für x

Fourier-Motzkin

- Es gilt:
  - max(untereSchranken) $\leq x \leq$ min(obere Schranken)
- Insbesondere
  - jede untere Schranke $\leq x \leq$ jede oberen Schranke
- Folglich sicher:
  - Kombiniere jede untere Schranke mit jeder oberen Schranke
  - Entferne x
  - Neues System von Ungleichungen, jetzt aber ohne x
  - Wenn das alte System eine Lösung hatte, dann auch das neue
  - leider nicht umgekehrt

Auflösung nach x

$x \geq 16-y$ (1)
$x \leq 7 - (7/4)\, y$ (2)
$x \leq 10 + (7/2)\, y$ (3)
$x \geq -9/2 + (3/2)\, y$ (4)

Alle möglichen Kombinationen:

$16-y \leq x \leq 7 - (7/4)\, y$
$16-y \leq x \leq 10 + (7/2)\, y$
$-9/2 + (3/2)\, y \leq x \leq 7 - (7/4)y$
$-9/2 + (3/2)\, y \leq x \leq 10 + (7/2)y$

Fourier-Motzkin

- Entferne x
  - Neues System, aber ohne x
- Wenn das alte System eine Lösung hatte, dann auch das neue
  - leider nicht umgekehrt
- Eliminiere so nacheinander alle Variablen

- Alle möglichen Kombinationen:

$$16-y \leq x \leq 7-(7/4)\,y$$
$$16-y \leq x \leq 10+(7/2)\,y$$
$$-9/2+(3/2)\,y \leq x \leq 7-(7/4)\,y$$
$$-9/2+(3/2)\,y \leq x \leq 10+(7/2)y$$

- Entferne x und vereinfache

$$y \leq -12$$
$$y \geq 4/3$$
$$y \leq 46/13$$
$$y \geq -29/4$$

Jetzt das Gleiche mit y:

$$4/3 \leq y \leq -12$$
$$4/3 \leq y \leq 46/13$$
$$-29/4 \leq y \leq -12$$
$$-29/4 \leq y \leq 46/13$$

Fourier-Motzkin

- Wenn das endgültige System keine Lösung hat, dann hat auch das ursprüngliche System keine Lösung
- Im Falle $Z = \mathbb{Q}$ oder $Z=\mathbb{R}$ gilt auch die Umkehrung
- Im Fall $Z=\mathbb{Z}$ gibt es Scheinlösungen

- Jetzt das Gleiche mit y:

  $4/3 \leq y \leq -12$

  $4/3 \leq y \leq 46/13$

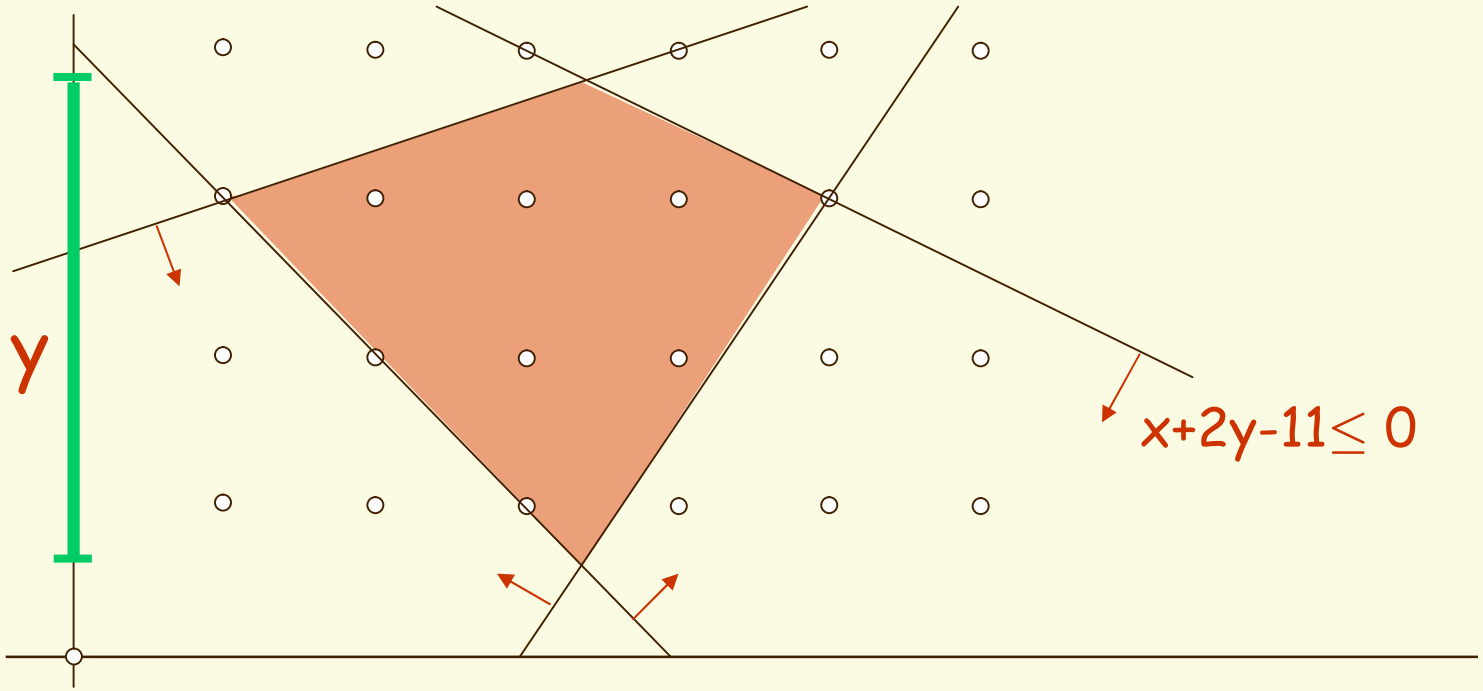
  $-29/4 \leq y \leq -12$

  $-29/4 \leq y \leq 46/13$
- Es folgt

  $4/3 \leq y \leq -12$
- Keine Lösung
- Daher hatte das ursprüngliche System keine Lösung

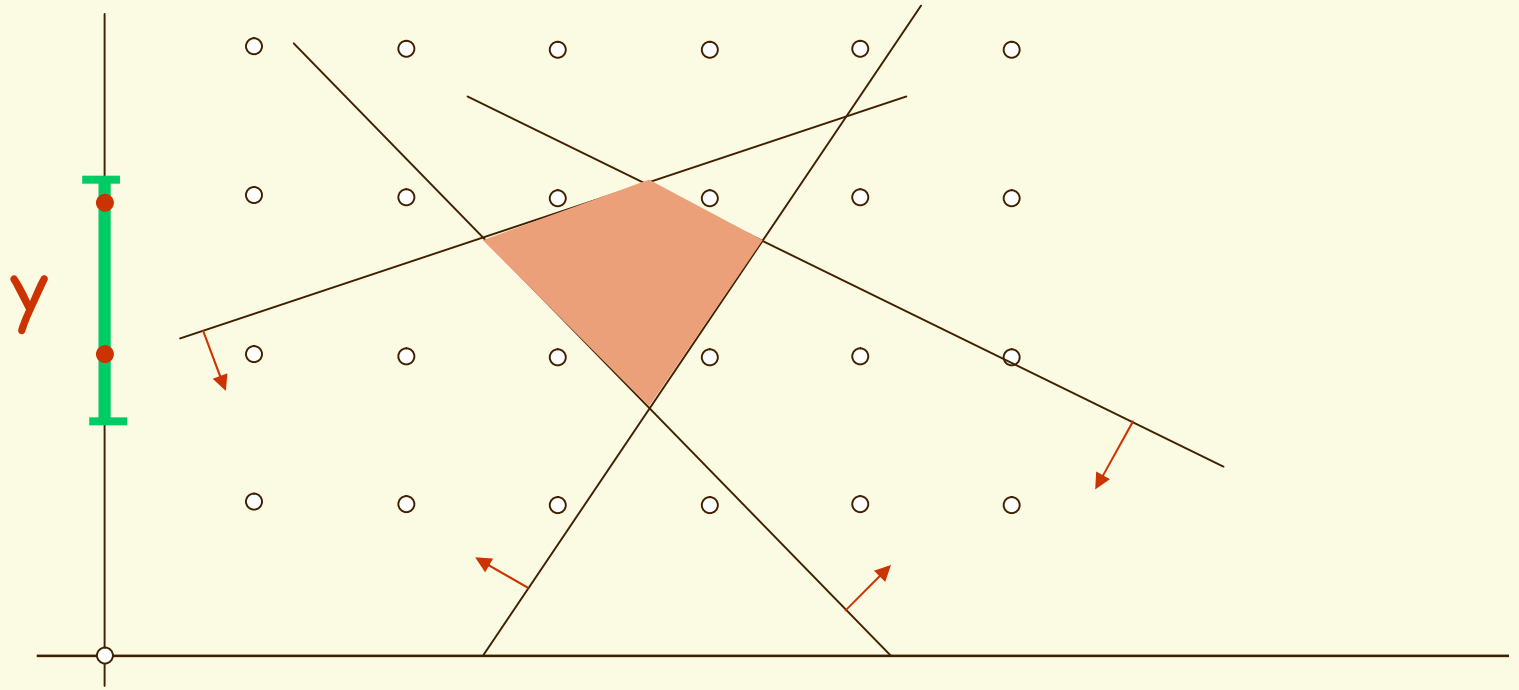
Geometrische Veranschaulichung

- Jede Ungleichung definiert eine Halbebene
- Konjunktion = Schnitt
- Resultat: konvexe Menge
- Elimination einer Variablen = Projektion auf Unterraum



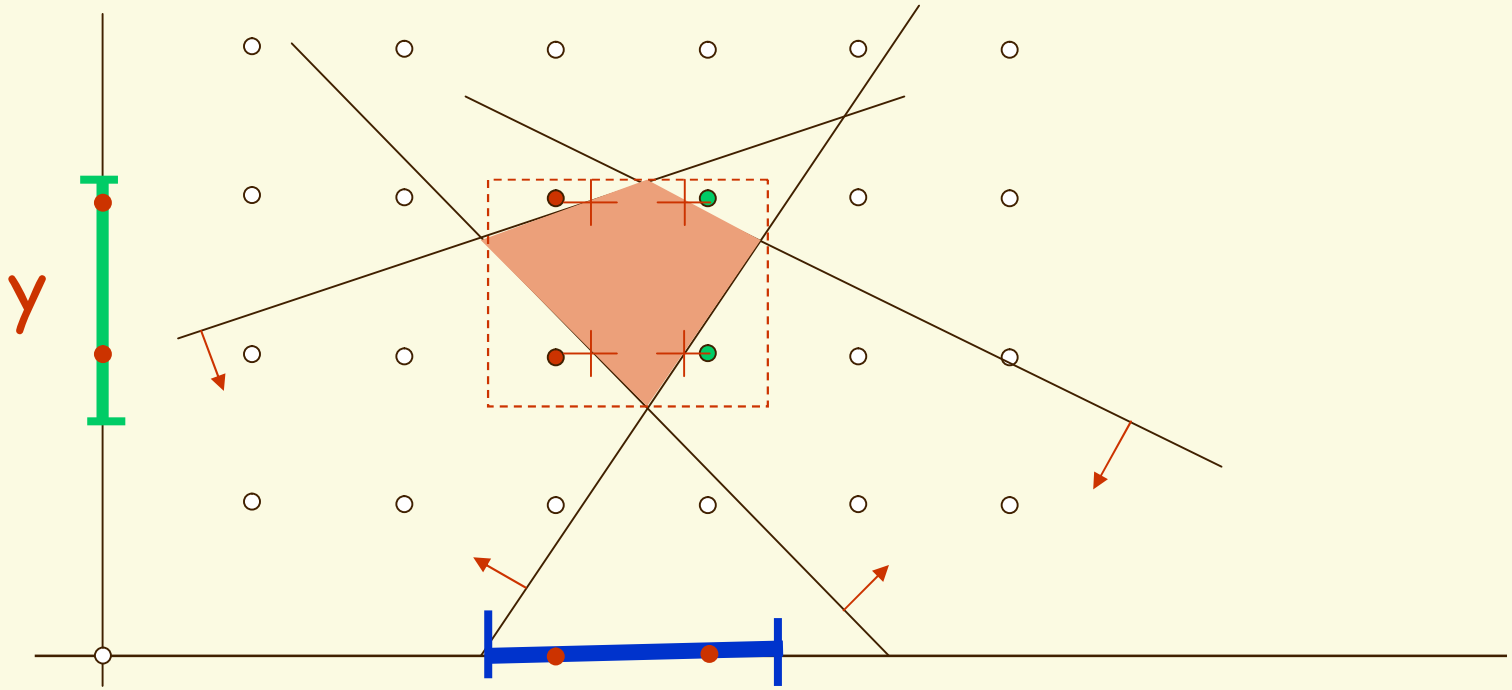
Falsch positive Lösungen

- Eine Lösung des endgültigen Systems garantiert keine Lösung des ursprünglichen Systems
  - falsch positive Lösungen
  - anders als im Falle $\mathbb{Q}$ oder $\mathbb{R}$



Testen falsch positiver Lösungen

```
FOR y := y_min TO y_max
  FOR x := ⌈max(untereSchranken(y))⌉ TO
           ⌊min(obereSchranken(y))⌋ DO
    Prüfe ob (x,y) die Ungleichungen erfüllt
```



Omega Test (W.Pugh)

- Theorie der Standardinterpretation $(\mathbb{Z},+,0,1,=,\leq)$
  - Gleichungen und Ungleichungen über $\mathbb{Z}$
  - Jeder Term kann in die Form $\Sigma_i a_i x_i$ gebracht werden,
  - jede atomare Aussage in: $\Sigma_i a_i x_i = a_0$ oder $\Sigma_i a_i x_i \leq a_0$

- 1. Schritt : Normalisierung
  - a_i durch $d:=ggT(a_1,\ldots,a_n)$ teilen
    - Falls a_0 nicht durch d teilbar $\Rightarrow$ unlösbar
    - Ansonsten
      - $\Sigma_i a_i x_i = a_0 \rightsquigarrow \Sigma_i (a_i/d) x_i = (a_0/d)$
      - $\Sigma_i a_i x_i \leq a_0 \rightsquigarrow \Sigma_i a_i x_i \leq \lfloor a_0/d \rfloor$
  - ab jetzt also $ggT(a_1,\ldots,a_n)=1$

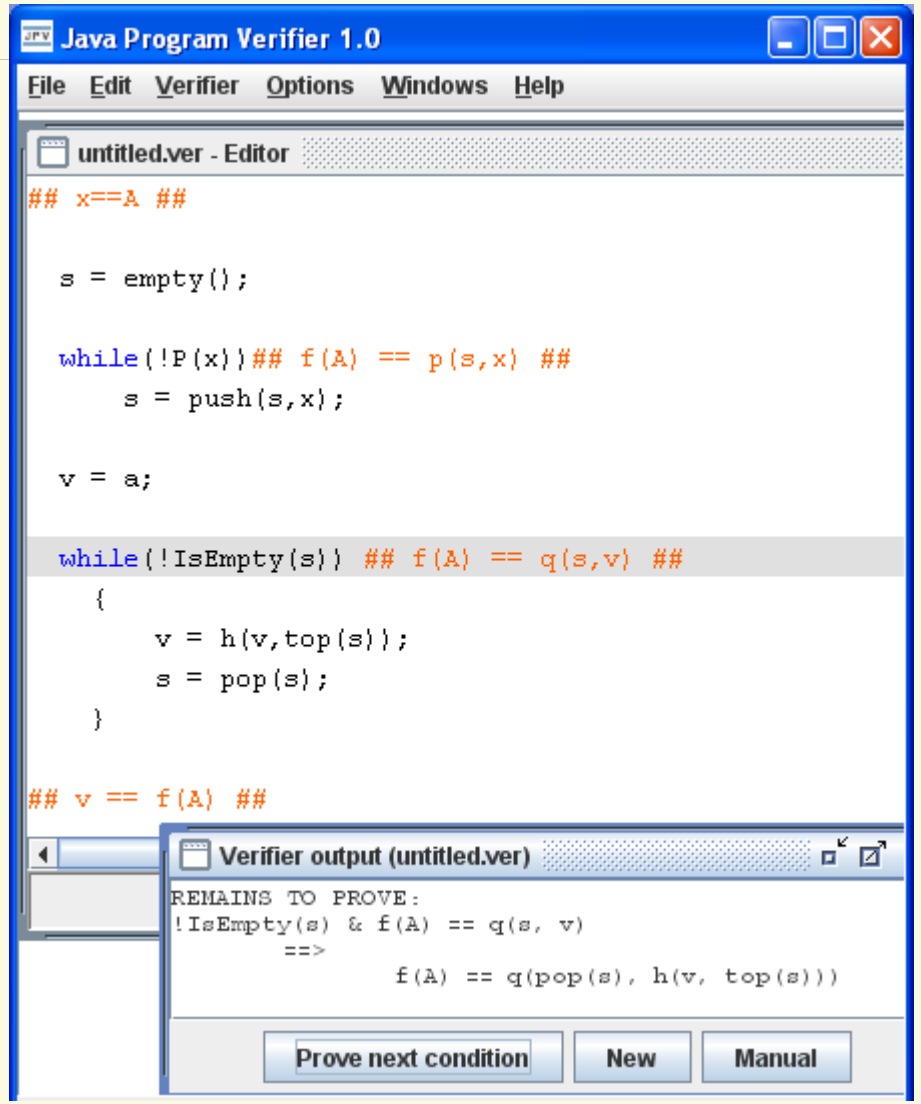
- 2. Schritt: Gleichungen lösen
  - Allgemeinste Lösung von $\Sigma_i a_i x_i = a_0$ bestimmen
  - In die Ungleichungen einsetzen

- 3. Schritt: Ungleichungen lösen
  - z.B. mit Fourier Motzkin

- W. Pugh beschreibt (und implementiert) Verbesserungen zu den einzelnen Schritten
- Erhältlich als „Omega-Test"
- W. Pugh: *A practical Algorithm for Exact Array data dependency analysis* Communications of the ACM, Vol. 35(8), 1992, p.102-114.

Funktionssymbole

- Neben Operationssymbolen (+,-,*,…) können bisher unbekannte „uninterpretierte“ Funktionen auftauchen
  - noch kein spezielles Entscheidungsverfahren vorgesehen
- Benutzerdefinierte Fkt.
  - fakultät
- Bibliotheksfunktionen
  - push, pop, store, lookup
- Hilfsfunktionen
  - z.B. abstrakte Invarianten



Uninterpretierte Funktionssymbole

- Sei Σ eine beliebige Signatur
  - $f_i : \tau_1 \times \tau_2 \times \ldots \times \tau_n \rightarrow \tau$ beliebige Funktionszeichen
  - kein Relationszeichen außer „=„
- $Th_{\Sigma,=}$ = die Menge aller offenen Aussagen, die in allen Σ-Strukturen gelten, z.B.:
  - $x=x$, (Reflexivität)
  - $x=y \rightarrow y=x$, (Symmetrie)
  - $x=y \wedge y=z \rightarrow x=z$ (Transitivität)
  - $x_1=y_1 \wedge \ldots \wedge x_n=y_n \rightarrow f(x_1,\ldots,x_n)=f(y_1,\ldots,y_n)$ (Kongruenz)
- Finde Entscheidungsverfahren basierend auf diesen Axiomen
  - Beispiel für ein Entscheidungsproblem:

    $a=g(a,b) \wedge f(f(a)) = f(g(a,b)) \rightarrow f(f(a)) = f(a)$

Kongruenz-Abschluss

- Repräsentiere Terme als Bäume
  - gemeinsame Teilbäume nur einmal
  - es entsteht ein Wald
- reflexive Relation R repräsentiere die gegebenen Gleichungen
  - wiederhole
    - Kongruenzabschluss:

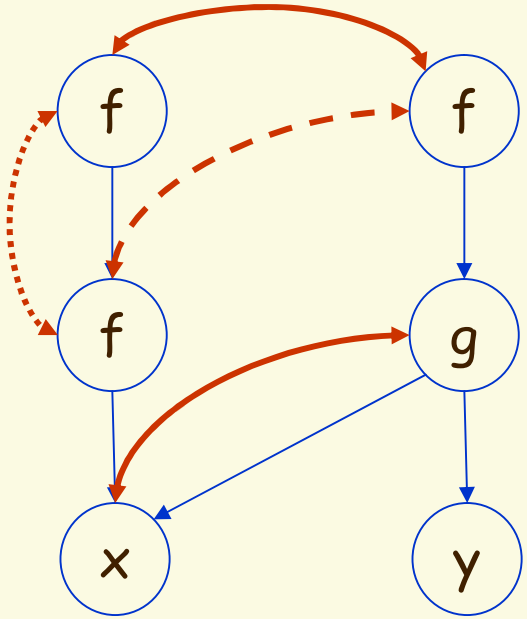
      $x_1 R y_1 \wedge \ldots \wedge x_n R y_n \rightarrow f(x_1,\ldots,x_n) R f(y_1,\ldots,y_n)$
    - Äquivalenzabschluss

      $R := (R \cup R^-)^*$

    bis nichts neues mehr entsteht
  - Prüfe, ob gesuchte Gleichung in R.

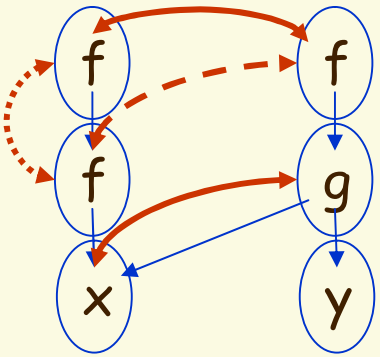
Beispiel: Congruence Closure

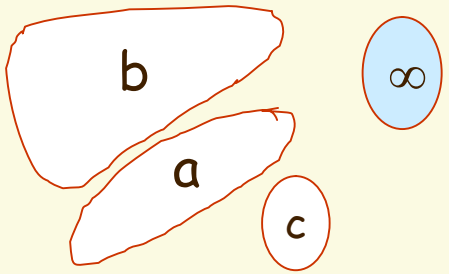
- Gegebenes Entscheidungsproblem:
  $\forall x,y.\ (\ x=g(x,y) \wedge f(f(x)) = f(g(x,y)) \rightarrow\ f(f(x)) = f(x)\ )$
- Konstruiere Wald aller Teilterme
- Füge die Prämissen hinzu ⟷
  (reflexive Pfeile hier nicht dargestellt)
- wiederhole
  - bilde Kongruenzabschluss ⟵ ⟶
  - symmetrisch-transitive Hülle
- lese ab, ob Konklusion besteht oder nicht ⟷



Korrektheit - Vollständigkeit

- Korrektheit ist offensichtlich
  - wenn aus den ursprünglichen Gleichheiten R eine neue Gleichheit folgt, dann muss diese in allen Modellen gelten, die R erfüllen
- Vollständigkeit
  - Wenn eine Gleichheit nicht folgt, dann gibt es ein Modell in der sie nicht gilt
  - Wähle
    - den konstruierten Wald
    - faktorisiere nach der erzeugten Kongruenz,
    - füge ein Element ∞ hinzu, das Ergebnis aller noch nicht definierten Operationen wird
  - Im Beispiel gilt:
    $\neg\ \forall x,y.(\ x=g(x,y) \wedge f(f(x)) = f(g(x,y))$
    $\rightarrow\ g(x,y) = f(x)\ \ )$





g(a,c) = a
f(a)=f(b)=b
sonst:
f(x)=g(x,y) = ∞

Funktionszeichen mit Axiomen

- Manche Funktionszeichen sind durch Axiome definiert
  - Stacks mit push, pop, top, isEmpty, empty
    - top(push(x,s)) = x
    - pop(push(x,s)) = y
  - Arrays mit write, read
    - Traditionell spezielle Syntax
      - a[i] = v; statt a = write(a,i,v);
      - x = a[i]; statt x = read(a,i)
    - Axiome z.B.
      - read(write(a,i,v),i) = v
      - $i \neq j \Rightarrow$ read(write(a,i,v),j) = read(a,j)
      - a=b $\Leftrightarrow \forall$ i. read(a,i) = read(b,i)
    - Es existiert ein Entscheidungsverfahren für offene Formeln

Termvereinfachung

- Eine Strategie die Gleichheit zweier Terme
  - $t_1 = t_2$
- zu entscheiden, ist, die Terme zu vereinfachen
  - $t_1 \rightsquigarrow s_1$, $t_2 \rightsquigarrow s_2$
- Für gewisse Theorien gilt dann
  - $t_1 = t_2 \Leftrightarrow s_1 \equiv s_2$
  - wobei $s_1 \equiv s_2$ bedeutet:
    - s_1 und s_2 sind syntaktisch gleich
- Beispiel Boolesche Algebren
  - $x \vee (\neg x \vee z) = z \vee (\neg z \vee x)$
  - $\rightsquigarrow x \vee z$ $\qquad\qquad$ $\rightsquigarrow z \vee x$
  - $\rightsquigarrow x \vee z$ $\qquad\qquad$ $\rightsquigarrow x \vee z$
  - $x \vee z \equiv x \vee z$

Normalformen

- Manche Theorien Th erlauben jeden Term t in eine äquivalente Normalform $\sigma(t)$ umzuwandeln, so dass gilt
  - $Th \models t = \sigma(t)$
  - $Th \models t_1 = t_2$ gdw. $\sigma(t_1) \equiv \sigma(t_2)$ ("$\equiv$" ist syntaktische Gleichheit)
- Eine Funktion σ, die jedem Term ihre Normalform zuordnet, heißt auch Kanonisierer (engl.: *canonizer*)
- <u>Beispiele</u>:
  - Boolesche Algebren
    - $\sigma(t)$:= DNF(t) oder alternativ: $\sigma(t)$ = KNF(t)
  - Presburger Arithmetik = Th(Z,+,-,0,1,=)
    - $\sigma(t) := a_0+a_1x_1+\ldots+a_nx_n$, $a_i \in Z$ (für festgelegte Variablenordnung)
  - Arithmetik = Th(Z,+,-,*,0,1,=)
    - $\sigma(t) := a_0+a_1x_1+\ldots+a_nx_n + a_{11}x_1x_1+\ldots+a_{1n}x_1x_n + \ldots$

Knuth-Bendix Methode

- Ist eine Theorie durch eine Menge von Gleichungen gegeben, so lässt sich ein canonizer oft auch durch die Knuth-Bendix-Methode finden
  - Zunächst muss eine syntaktische Ordnung $\leq$ (einfacher als) auf den Termen definiert werden
  - Aus jeder Gleichung $t_1=t_2$ wird eine gerichtete Gleichung ($t_1 \geq t_2$ oder $t_2 \geq t_1$) geschrieben.
  - Die Idee ist, die Gleichungen nur von links nach rechts (also vereinfachend) anzuwenden
  - Es gibt aber ein Problem, wenn ein Term auf mehrere Weisen vereinfacht werden kann:

Beispiel: Gruppen

$x \circ (y \circ z) \geq (x \circ y) \circ z$
$x^{-1} \circ x \geq e$
$e \circ x \geq x$

Aber:

$x \circ (y^{-1} \circ y)$

vereinfacht zu

$(x \circ y^{-1}) \circ y$

und zu

x

Beide Terme können zunächst nicht weiter vereinfacht werden

Knuth-Bendix completion

- Zu Gleichungen $l_1 \geq r_1$ und $l_2 \geq r_2$ prüfe, ob es ein t gibt, der zu verschiedenen Normalformen reduziert
  - (Man gewinnt alle in Frage kommenden Terme t durch Unifikation von l_i mit nichttrivialen Teiltermen von l_{1-i})
- Füge die entstandenen Normalformen als neue Gleichung hinzu
- Falls das Verfahren terminiert, liefert es einen Kanonisierer
- Es gibt viele Vorschläge für geeignete Ordnungen
- Schwierigkeiten u.a. mit Kommutativität

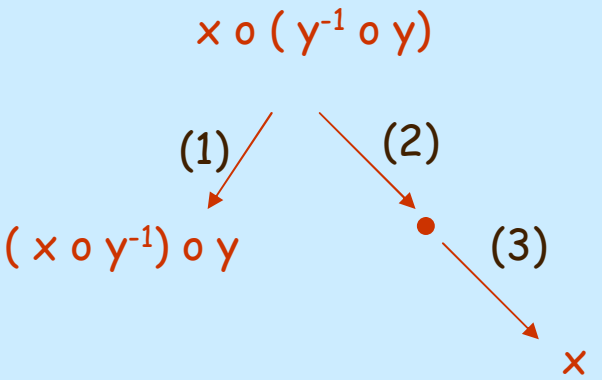
  $x \circ y \leq y \circ x$???

Beispiel: Gruppen

(1) $x \circ (y \circ z) \geq (x \circ y) \circ z$

(2) $x^{-1} \circ x \geq e$

(3) $x \circ e \geq x$



Neue Gleichung

(4) $(x \circ y^{-1}) \circ y \geq x$

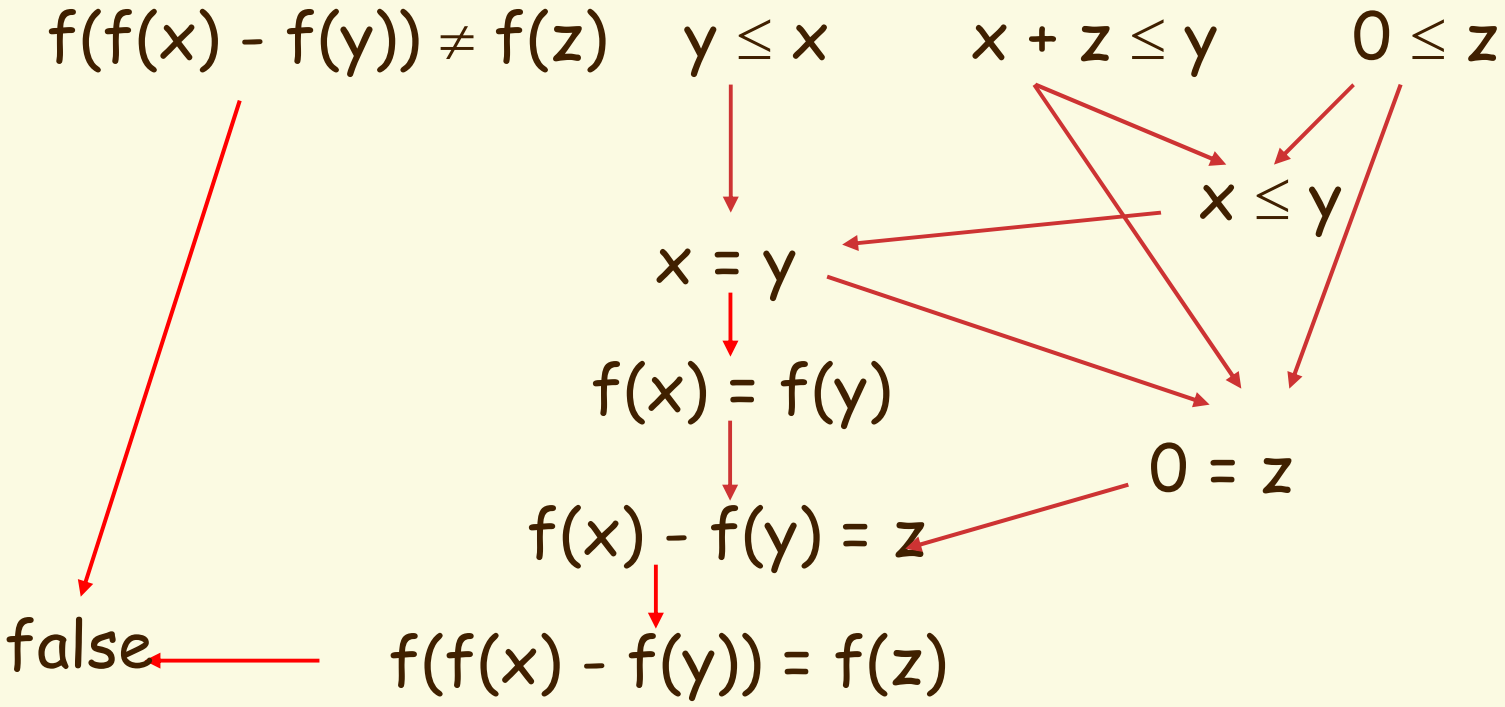
Verbindung von Entscheidungsprozeduren

- Realistische Entscheidungsprobleme beinhalten Formeln mehrerer Theorien

- Beispiele
  - $f(y-1)-1=y+1,\ f(x)+1=x-1,\ x+1=y \vdash \text{false}$
  - $f(f(x)-f(y)) \neq f(z),\ y \leq x,\ y \geq x+z,\ z \geq 0 \vdash \text{false}$
  - $x+2=y \vdash f(a[x:=3][y-2])=f(y-x+1)$

- In den Beispielen wurden kombiniert:
  - Th($\mathbb{Z}$,+,=) und uninterpretierte Funktionssymbole
  - Th($\mathbb{Z}$,+,$\leq$, =) und uninterpretierte Funktionssymbole
  - Th($\mathbb{Z}$, +,=) und uninterpretierte Funktionssymbole und Arrays

Gleichheit kombiniert mit Arithmetik



Situation

- Gegeben (mindestens) zwei Signaturen Σ_1 und Σ_2
  - z.B. Σ_1 = (f ; =) und Σ_2 = (+,-,0 ; =)
  - definieren logische Sprachen $\mathcal{L}_1 = \mathcal{L}(\Sigma_1)$ und $\mathcal{L}_1 = \mathcal{L}(\Sigma_1)$
- Signaturen seien disjunkt – bis auf Gleichheit `=´.
  - $\mathcal{L}_1 \cap \mathcal{L}_2$ enthält nur Formeln, mit =, $\neq$ und den logischen Operatoren
- Gegeben eine Theorie (Menge von Formeln) T_1 in der Sprache $\mathcal{L}_1$ und eine Theorie T_2 in $\mathcal{L}_2$
  - Beispiel: T_1={ x=y$\Rightarrow$ f(x)=f(y), …}, T_2={ x-x=0, … }
- Gegeben eine Formel Φ aus $\mathcal{L}_1 \cup \mathcal{L}_2$
- Zeige $\Phi \models \bot$ unter Zuhilfenahme von Entscheidungsprozeduren für T_1 und für T_2

Ackermann's Idee

- Entkopple die Formel Φ durch Einführung zusätzlicher Variablen
- Finde
  - Formeln $\Phi_1 \in \mathcal{L}(\Sigma_1)$ und $\Phi_2 \in \mathcal{L}(\Sigma_2)$
  - so das $\Phi \vDash \bot \Leftrightarrow \Phi_1 \wedge \Phi_2 \vDash \bot$
- Aber beachte: Φ_1 und Φ_2 haben zusätzliche Variablen

Entkopplung

- Führe neue Variablen für Teilterme ein.

$$f(\underbrace{f(x)}_{u}-\underbrace{f(y)}_{v}) \neq f(z),\ y \leq x,\ y \geq x+z,\ z \geq 0 \vdash \text{false}$$

- Neue Variablen u,v, neue Gleichungen
  - $u=f(x),\ v=f(y),\ f(\overbrace{u-v}^{w}) \neq f(z),\ y \leq x,\ y \geq x+z,\ z \geq 0$
  - Noch eine neue Variable w:
    - $u=f(x),\ v=f(y),\ w=u-v,\ f(w) \neq f(z),\ y \leq x,\ y \geq x+z,\ z \geq 0$
- Entkoppelte Formeln
  - $\Phi_1 = w=u-v,\ y \leq x,\ y \geq x+z,\ z \geq 0 \quad \in \mathcal{L}(\Sigma 2)$
  - $\Phi_2 = u=f(x),\ v=f(y),\ f(w) \neq f(z) \quad \in \mathcal{L}(\Sigma_1)$
  - Ursprüngliche Formel erfüllbar $\Leftrightarrow$ $\Phi_1 \wedge \Phi_2$ erfüllbar

Nelson-Oppen

- Situation:
  - Gegeben Entscheidungsverfahren D_1 für Th_1 und D_2 für Th_2
  - Gegeben Formel $\Phi \in \mathcal{L}(\Sigma_1) \cup \mathcal{L}(\Sigma_2)$,
  - Entkoppele in Formeln $\Phi_1 \in \mathcal{L}(\Sigma_1)$ und $\Phi_2 \in \mathcal{L}(\Sigma_2)$, so dass
    - $\Phi \models \bot \Leftrightarrow \Phi_1 \wedge \Phi_2 \models \bot$
  - Φ_1 und Φ_2 haben zusätzliche Variablen $x_1,\ldots,x_n$
- Verwende D_1 um aus Φ_1 eine Gleichung $x_i=x_j$ (oder Ungleichung $x_i \neq x_j$) herzuleiten
- Verwende D_2 um aus $\Phi_2 \cup \{x_i=x_j\}$ eine weitere Gleichung (Ungleichung) herzuleiten
- u.s.w bis Widerspruch gefunden

NO-Beispiel: 1. Entkopplung

$(x_1 \leq x_2) \wedge (x_2 \leq (x_1 + car(cons(0,x_1)))) \wedge p(h(x_1) - h(x_2)) \wedge \neg p(0)$

- Entkopplung durch Einführung von Variablen:

$(x_1 \leq x_2) \wedge (x_2 \leq x_1 + a_1) \wedge p(a_2) \wedge \neg p(a_5) \wedge$
$a_1 = car(cons(a_5, x_1)) \wedge$
$a_2 = a_3 - a_4 \quad \wedge$
$a_3 = h(x_1) \quad \wedge$
$a_4 = h(x_2) \quad \wedge$
$a_5 = 0$

(aus: Kroenig,Strichmann)

Nelson-Oppen-Beispiel

$(x_1 \leq x_2) \wedge (x_2 \leq (x_1 + car(cons(0,x_1)))) \wedge p(h(x_1) - h(x_2)) \wedge \neg p(0)$

Arithmetik	EUF	Listen
$x_1 \leq x_2$ $x_2 \leq x_1 + a_1$ $a_2 = a_3 - a_4$ $a_5 = 0$	$a_3 = h(x_1)$ $a_4 = h(x_2)$ $p(a_2)$ $\neg p(a_5)$	$a_1 = car(cons(a_5,x_1))$

Nelson-Oppen-Beispiel

$(x_1 \leq x_2) \wedge (x_2 \leq (x_1 + car(cons(0,x_1)))) \wedge p(h(x_1) - h(x_2)) \wedge \neg p(0)$

Arithmetik	EUF	Listen
$x_1 \leq x_2$ $x_2 \leq x_1 + a_1$ $a_2 = a_3 - a_4$ $a_5 = 0$	$a_3 = h(x_1)$ $a_4 = h(x_2)$ $p(a_2)$ $\neg p(a_5)$	$a_1 = car(cons(a_5,x_1))$ $\boxed{a_1 = a_5}$ Gleichheit gefunden

Nelson-Oppen-Beispiel

$(x_1 \leq x_2) \wedge (x_2 \leq (x_1 + car(cons(0,x_1)))) \wedge p(h(x_1) - h(x_2)) \wedge \neg p(0)$

Arithmetik	EUF	Listen
$x_1 \leq x_2$	$a_3 = h(x_1)$	$a_1 = car(cons(a_5,x_1))$
$x_2 \leq x_1 + a_1$	$a_4 = h(x_2)$	
$a_2 = a_3 - a_4$	$p(a_2)$	
$a_5 = 0$	$\neg p(a_5)$	
$a_1 = a_5$	$a_1 = a_5$	$\boxed{a_1 = a_5}$ Gleichheit propagieren

Nelson-Oppen-Beispiel

$(x_1 \leq x_2) \wedge (x_2 \leq (x_1 + car(cons(0,x_1)))) \wedge p(h(x_1) - h(x_2)) \wedge \neg p(0)$

Arithmetik	EUF	Listen
$x_1 \leq x_2$	$a_3 = h(x_1)$	$a_1 = car(cons(a_5,x_1))$
$x_2 \leq x_1 + a_1$	$a_4 = h(x_2)$	
$a_2 = a_3 - a_4$	$p(a_2)$	
$a_5 = 0$	$\neg p(a_5)$	
$a_1 = a_5$	$a_1 = a_5$	$a_1 = a_5$
$x_1 = x_2$		
neue Gleichheit gefunden	etc...	

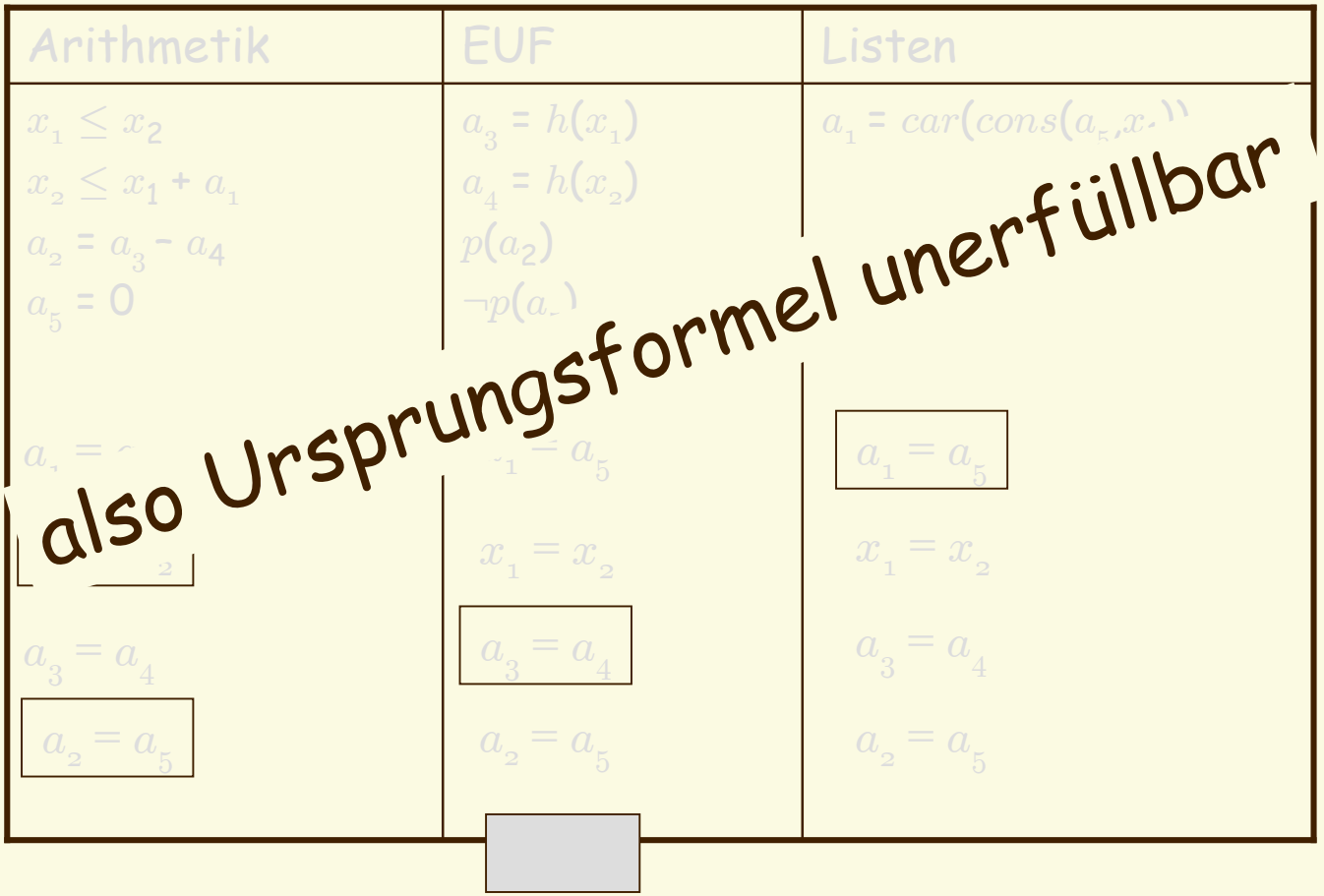
Nelson-Oppen-Beispiel

$(x_1 \leq x_2) \wedge (x_2 \leq (x_1 + car(cons(0,x_1)))) \wedge p(h(x_1) - h(x_2)) \wedge \neg p(0)$

Arithmetik	EUF	Listen
$x_1 \leq x_2$	$a_3 = h(x_1)$	$a_1 = car(cons(a_5,x_1))$
$x_2 \leq x_1 + a_1$	$a_4 = h(x_2)$	
$a_2 = a_3 - a_4$	$p(a_2)$	
$a_5 = 0$	$\neg p(a_5)$	
$a_1 = a_5$	$a_1 = a_5$	$a_1 = a_5$
$x_1 = x_2$	$x_1 = x_2$	$x_1 = x_2$
$a_3 = a_4$	$a_3 = a_4$	$a_3 = a_4$
$a_2 = a_5$	$a_2 = a_5$	$a_2 = a_5$
	False	

Nelson-Oppen-Beispiel

$(x_1 \leq x_2) \wedge (x_2 \leq (x_1 + car(cons(0,x_1)))) \wedge p(h(x_1) - h(x_2)) \wedge \neg p(0)$



Craig's Interpolation Lemma

- Lemma(Craig) Seien $\Phi_1 \in \mathcal{L}(\Sigma_1)$ und $\Phi_2 \in \mathcal{L}(\Sigma_2)$.
- Wenn $\Phi_1 \models \Phi_2$ dann gibt es eine Formel $\varphi \in \mathcal{L}(\Sigma_1) \cap \mathcal{L}(\Sigma_2)$, so dass
  - $\Phi_1 \models \varphi$
  - $\varphi \models \Phi_2$
- Ein solches φ heißt Interpolationsformel.
- Folgerung: In unserem Falle, $\mathcal{L}(\Sigma_1) \cap \mathcal{L}(\Sigma_2) = \emptyset$ (bis auf Gleichheitsrelation), können in φ nur Gleichheiten und Ungleichheiten auftauchen.
- Falls Φ_1, Φ_2 offene Formeln sind, kann φ als offene Formel gewählt werden

Craig's Lemma (für offene Formeln)

Induktion über die Länge des Beweises.

Der letzte Schritt im Beweis von $\Gamma \vdash \Delta$ sei:

- Axiom, weil $\phi \in \Gamma$ und $\phi \in \Delta$
  - Dann ist ϕ Interpolant, denn $\Gamma \vdash \phi$ und $\phi \vdash \Delta$
- $\wedge$-rechts:
  - $\dfrac{\Gamma \vdash P_1, \Delta \quad \Gamma \vdash P_2, \Delta}{\Gamma \vdash P_1 \wedge P_2, \Delta}$
  - Nach Vorrauss. ex. Interpolanten ϕ_i für $\Gamma \vdash P_i, \Delta$
    - also $\Gamma \vdash \phi_1$ und $\phi_1 \vdash P_1, \Delta$
    - und $\Gamma \vdash \phi_2$ und $\phi_2 \vdash P_2, \Delta$
  - dann ist $\phi_1 \wedge \phi_2$ Interpolant für $\Gamma \vdash P_1 \wedge P_2, \Delta$
- etc.

Folgerung aus Craig's Lemma

- Seien $\Phi_1 \in \mathcal{L}_1$, $\Phi_2 \in \mathcal{L}_2$ offene Formeln, $\Phi_1 \wedge \Phi_2$ widersprüchlich in $Th_1 \cup Th_2$. Dann gibt es endliche Teilmengen $T_1 \subseteq Th_1$ und $T_2 \subseteq Th_2$ mit
  - $T_1, \Phi_1, T_2, \Phi_2 \vdash \bot$.
- Also $T_1, \Phi_1 \vdash \neg T_2, \neg \Phi_2$.

- Nach Craig gibt es Interpolanten ϕ in $\mathcal{L}_1 \cap \mathcal{L}_2$:
  - $T_1, \Phi \vdash \phi$ und $\phi \vdash \neg T_2, \neg \Phi_2$ d.h.

  - $T_1, \Phi_1 \vdash \phi$ und $T_2, \Phi_2, \phi \vdash \bot$

- ϕ offene Formel in $\mathcal{L}(=)$ – also Disjunktion von Konjunktionen von Variablengleichungen und -ungleichungen.

2. Beispiel

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

- 1. Entkopplung:

$(x_2 \geq x_1) \wedge (x_1 - x_3 \geq x_2) \wedge (x_3 \geq 0) \wedge f(a_1) \neq f(x_3) \wedge$
$a_1 = a_2 - a_3 \quad \wedge$
$a_2 = f(x_1) \quad \wedge$
$a_3 = f(x_2)$

2. $T_1, \Phi_1 \vdash \phi$ und $T_2, \Phi_2, \phi \vdash \bot$

Mehrere konsekutive Interpolanten ϕ, ϕ', ϕ''

Arithmetik	EUF
$x_2 \geq x_1$ $x_1 - x_3 \geq x_2$ $x_3 \geq 0$ $a_1 = a_2 - a_3$ } Φ_1	$f(a_1) \neq f(x_3)$ $a_2 = f(x_1)$ $a_3 = f(x_2)$ } Φ_2

2. $T_1, \Phi_1 \vdash \phi$ und $T_2, \Phi_2, \phi \vdash \bot$

Mehrere konsekutive Interpolanten ϕ, ϕ', ϕ''

Arithmetik	EUF
$x_2 \geq x_1$ $x_1 - x_3 \geq x_2$ $x_3 \geq 0$ $a_1 = a_2 - a_3$ } Φ_1 $x_3 = 0$ $x_1 = x_2$ } ϕ	$f(a_1) \neq f(x_3)$ $a_2 = f(x_1)$ $a_3 = f(x_2)$ } Φ_2

2. $T_1, \Phi_1 \vdash \phi$ und $T_2, \Phi_2, \phi \vdash \bot$

Mehrere konsekutive Interpolanten ϕ, ϕ', ϕ''

Arithmetik		EUF	
$x_2 \geq x_1$ $x_1 - x_3 \geq x_2$ $x_3 \geq 0$ $a_1 = a_2 - a_3$	Φ_1	$f(a_1) \neq f(x_3)$ $a_2 = f(x_1)$ $a_3 = f(x_2)$	Φ_2
$x_3 = 0$ $x_1 = x_2$	ϕ	$x_3 = 0$ $x_1 = x_2$	ϕ
		$a_2 = a_3$	ϕ'

2. $T_1, \Phi_1 \vdash \phi$ und $T_2, \Phi_2, \phi \vdash \bot$

Mehrere konsekutive Interpolanten ϕ, ϕ', ϕ''

Arithmetik		EUF	
$x_2 \geq x_1$ $x_1 - x_3 \geq x_2$ $x_3 \geq 0$ $a_1 = a_2 - a_3$	Φ_1	$f(a_1) \neq f(x_3)$ $a_2 = f(x_1)$ $a_3 = f(x_2)$	Φ_2
$x_3 = 0$	ϕ	$x_3 = 0$	ϕ
$x_1 = x_2$		$x_1 = x_2$	
$a_2 = a_3$	ϕ'	$a_2 = a_3$	ϕ'
$a_1 = 0$	ϕ''	$a_1 = 0$	ϕ''
		False	

Es geht nicht immer gut - Beispiel

- Betrachte: $\phi: 1 \leq x \wedge x \leq 2 \wedge p(x) \wedge \neg p(1) \wedge \neg p(2) , \; x \in \mathbb{Z}$

Arithmetik über $\mathbb{Z}$	Uninterpret.Prädikate
$1 \leq x$ $x \leq 2$	$p(x)$ $\neg p(1)$ $\neg p(2)$

- Jede der Einzeltheorien ist erfüllbar
- Keine impliziert eine Gleichung
- Dennoch ist ϕ unerfüllbar!

Es geht nicht immer gut - Beispiel

- Betrachte: $\phi: 1 \leq x \wedge x \leq 2 \wedge p(x) \wedge \neg p(1) \wedge \neg p(2) , \; x \in \mathbb{Z}$

Arithmetik über $\mathbb{Z}$	Uninterpret.Prädikate
$1 \leq x$ $x \leq 2$ $x = 1 \vee x=2$	$p(x)$ $\neg p(1)$ $\neg p(2)$

Disjunktion von Gleichungen

- Jede der Einzeltheorien ist erfüllbar
- Keine impliziert eine Gleichung
- Dennoch ist ϕ unerfüllbar!

Konvexität

- Eine Theorie T heisst *konvex*, falls für jede Menge von Literalen L und je zwei Variablengleichungen $x_1=y_1$, $x_2=y_2$ gilt:
  - **Wenn $T, L \vdash x_1 = y_1 \vee x_2 = y_2$, dann T , $L \vdash x_1 = y_1$ oder $T, L \vdash x_2 = y_2$**
- Beispiel:
  - Th($\mathbb{R}$,0,1,+,-,=) ist konvex
    - Lösungen von Gleichungen sind affine Unterräume
    - Ist ein affiner UR enthalten in einer Vereinigung von zwei UR, dann auch in einem davon.

andererseits

- Th($\mathbb{Z}$,0,1,+,-, $\leq$,=) ist nicht konvex
  - denn z.B. $1 \leq x \wedge x \leq 2 \rightarrow x = 1 \vee x=2$
    - aber nicht $1 \leq x \wedge x \leq 2 \rightarrow x = 1$
    - und nicht $1 \leq x \wedge x \leq 2 \rightarrow x = 2$

NO für nichtkonvexe Theorien

- Propagiere Disjunktionen von Variablengleichungen
- Führt zu Fallunterscheidung in der anderen Theorie
  - $\Rightarrow$ Komplexität steigt

Arithmetik über $\mathbb{Z}$	Uninterpretierte Prädikate
$1 \leq x$ $x \leq 2$ $x = 1 \vee x = 2$	$p(x)$ $\neg p(1) \wedge \neg p(2)$ $x = 1 \vee x = 2$ *Split!* $x = 1$ *False* | $x = 2$ *False*

Der nichtkonvexe Fall

1. Zerlege ϕ in reine Formeln: $\phi' := F_1 \wedge \ldots \wedge F_n$.

   d.h. $F_1 \in \mathcal{L}(L_i)$

2. Falls eines der F_i unerfüllbar ist, return `unerfüllbar'.

3. Falls eines der F_i eine neue Variablengleichheit impliziert, propagiere diese in die anderen Theorien und goto 2.

4. Falls $\exists i.\ F_i \rightarrow (x_1 = y_1 \vee \ldots \vee x_k = y_k)$ aber $\forall j\ F_i \not\rightarrow x_j = y_j$, prüfe rekursiv $\phi' \wedge x_1 = y_1$, ... , $\phi' \wedge x_k = y_k$.
   Wenn eine davon erfüllbar ist, return 'erfüllbar', ansonsten return 'unerfüllbar'.

5. Return `erfüllbar'.

Offene Formeln in $\mathcal{L}(=)$

- Offene Formeln Δ in $\mathcal{L}(=)$ sind
- Disjunktionen von
  - Konjunktionen von Gleichungen $x_i=x_j$ und Ungleichungen $x_i \neq x_j$ von Variablen
    - $x_{i1}=y_{i1} \wedge \ldots x_{in}= y_{in} \wedge x_{in+1} \neq y_{in+1} \wedge \ldots \wedge x_{ik} \neq y_{ik}$.
  - Jede solche beschreibt eine Äquivalenzrelationen δ auf den Variablen.
  - δ beschreibt, welche Variablen in einem Modell zu identifizieren sind, und welche nicht.
  - also $\Delta = \delta_1 \vee \ldots \vee \delta_k$, für geeignete δ_k.

Existenz gemeinsamer Modelle

- Frage: Ist Φ erfüllbar in $Th_1 \cup Th_2$?

- Gegeben Sat_1, Sat_2 Entscheidungsprozeduren für Th_1, Th_2
- Mit zusätzlicher Variablenmenge V zerlege Φ in
  - $\Phi_1 \in \mathcal{L}(\Sigma_1)$ und $\Phi_2 \in \mathcal{L}(\Sigma_2)$.

- Wähle eine Äquivalenzrelation δ auf V
  - Prüfe mittels Sat_1, ob $\Phi_1 \wedge \delta$ erfüllbar
  - Prüfe mittels Sat_2, ob $\Phi_2 \wedge \delta$ erfüllbar

- Wenn für kein ~ erfolgreich, ist Φ nicht erfüllbar

- Wenn für ein δ
  - $\Phi_1 \wedge \delta$ erfüllbar
  - $\Phi_2 \wedge \delta$ erfüllbar
  - Ist dann Φ erfüllbar ?

Die Voraussetzung

- $\Phi_1 \wedge \delta$ erfüllbar in Th_1 heißt:
  - Es gibt ein Modell M_1 und eine Belegung der Variablen,

    so dass alle Formeln aus $\Phi_1 \wedge \delta$ wahr sind

- $\Phi_2 \wedge \delta$ erfüllbar in Th_2
  - Es gibt ein Modell M_2 und eine Belegung der Variablen,

    so dass alle Formeln aus $\Phi_1 \wedge \Delta_{\sim}$ wahr sind

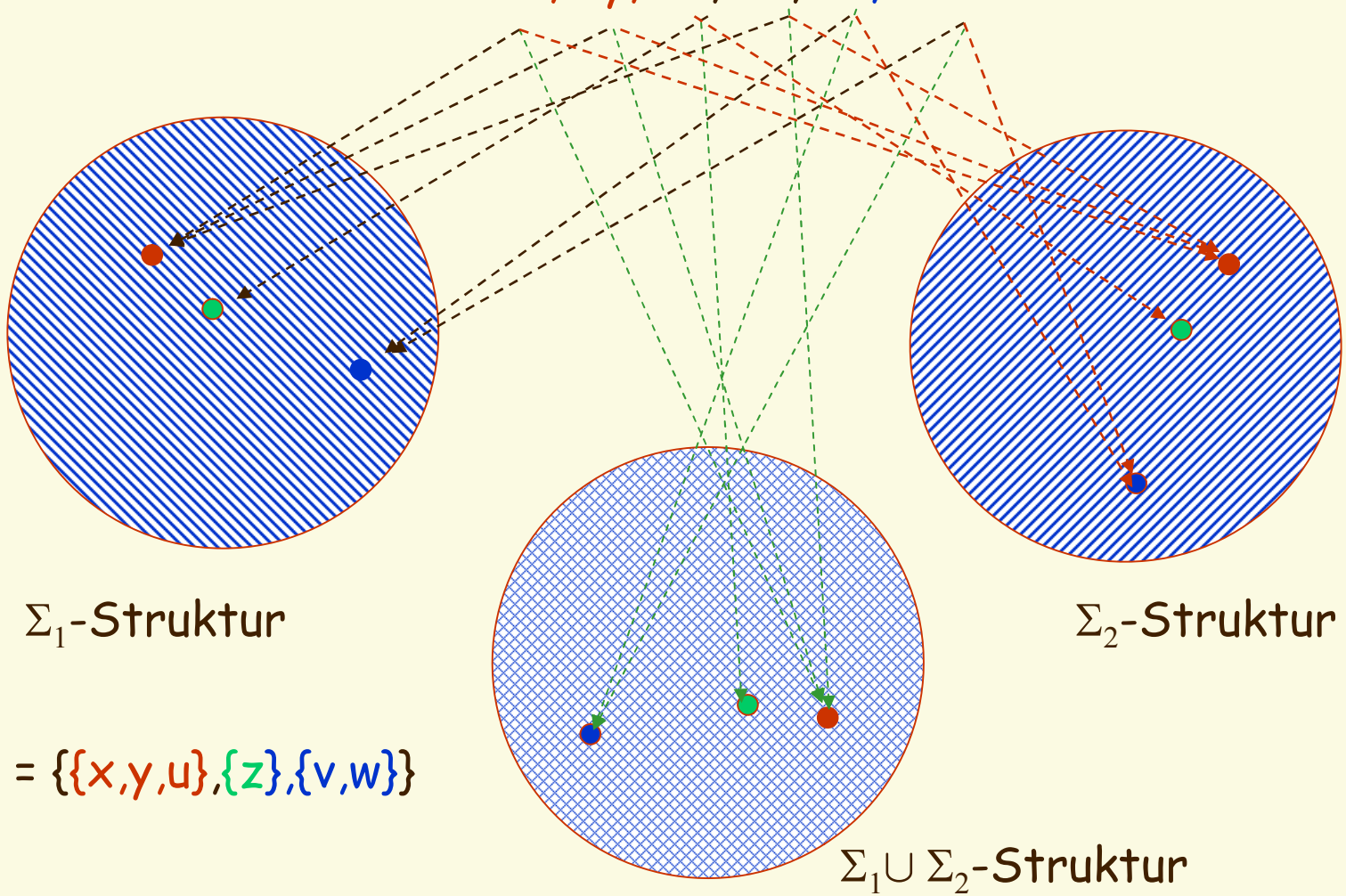
- Wir brauchen aber M_1 und M_2 mit $|M_1| = |M_2|$?

- In dem Fall kann man auf der gleichen Grundmenge M
  - eine Σ_1-Struktur M_1 und
  - eine Σ_2-Struktur M_2 definieren,
  - so dass mit der gleichen Belegung der gemeinsamen Variablen Φ erfüllt ist

- Im Allgemeinen kann man M abzählbar unendlich wählen

Modelle auf gemeinsamer Menge

- Variablen: x, y, z, u, v, w



δ = {{x,y,u},{z},{v,w}}

Stabil unendlich

- Zu zwei Theorien müssen wir ein Modell auf einer gleich großen Grundmenge finden.
- Eine Theorie T heißt stabil unendlich, wenn zu jeder Aussage ϕ gilt:

  Wenn $T \cup \{\phi\}$ erfüllbar ist, dann auch in einem unendlichen Modell.
- Der Satz von Löwenheim-Skolem besagt: *Wenn eine Menge von Aussagen ein unendliches Modell hat, dann auch ein abzählbares.*
- Sind also T_1 und T_2 stabil unendlich, und sind $T_1 \cup \{\delta\}$ und $T_2 \cup \{\delta\}$ erfüllbar, dann auch auf einer gemeinsamen abzählbar unendlichen Menge.

Konvex impliziert stabil unendlich

- Jede nichttriviale konvexe Theorie ist stabil unendlich.
  (*Nichttrivial* soll hier heißen: Es existiert ein Modell mit **mehr als einem** Element)
- Beweis: Für jede Zahl $n \in \mathbb{N}$ sei ϕ_n die Formel $\exists x_1,\dots,x_n. \wedge_{i \neq j} x_i \neq x_j$,
  die besagt, dass es mindestens n verschiedene Elemente gibt.
- Angenommen, T sei nicht stabil unendlich. Dann gibt es eine Formel ψ,
  so dass $T \cup \{\psi\}$ kein unendliches Modell hat.
- Damit ist die Menge $T \cup \{\psi\} \cup \{ \phi_n \mid n \in \mathbb{N} \}$ widersprüchlich.
- Damit ist aber schon eine endliche Teilmenge widersprüchlich. Es gibt also
  ein m, so dass jedes Modell, von $T \cup \{\psi\}$ maximal m Elemente besitzt.
- Es folgt: $T \cup \{ \psi \} \models \vee \{x_i = x_j \mid i \neq j, i,j \leq m \}$
- Mit Konvexität folgt: $T \cup \{ \psi \} \models \{ x = y \}$, also $T \cup \{\psi\}$ hat kein nichttriviales
  Modell.
- Widerspruch

Nicht stabil unendliche Theorien

- Betrachte Theorie $\mathcal{T}_1$:
  - Σ_1: Eine Funktion f,
  - Axiome, die nur 2 verschiedene Werte erlauben
- Eine Theorie $\mathcal{T}_2$:
  - Σ_2: Eine Function g,
  - Domain: $\mathbb{N}$

Die kombinierte Theorie $\mathcal{T}_1 \oplus \mathcal{T}_2$ hat die Vereinigung der Axioms.
Insbesondere nur höchstens zweielementige Modelle

Also ist die folgende Formel unerfüllbar:

$$\phi\colon\ f(x_1) \neq f(x_2) \ \wedge\ g(x_1) \neq g(x_3) \ \wedge\ g(x_2) \neq g(x_3)$$

aber

- Nelson-Oppen liefert keine Gleichungen

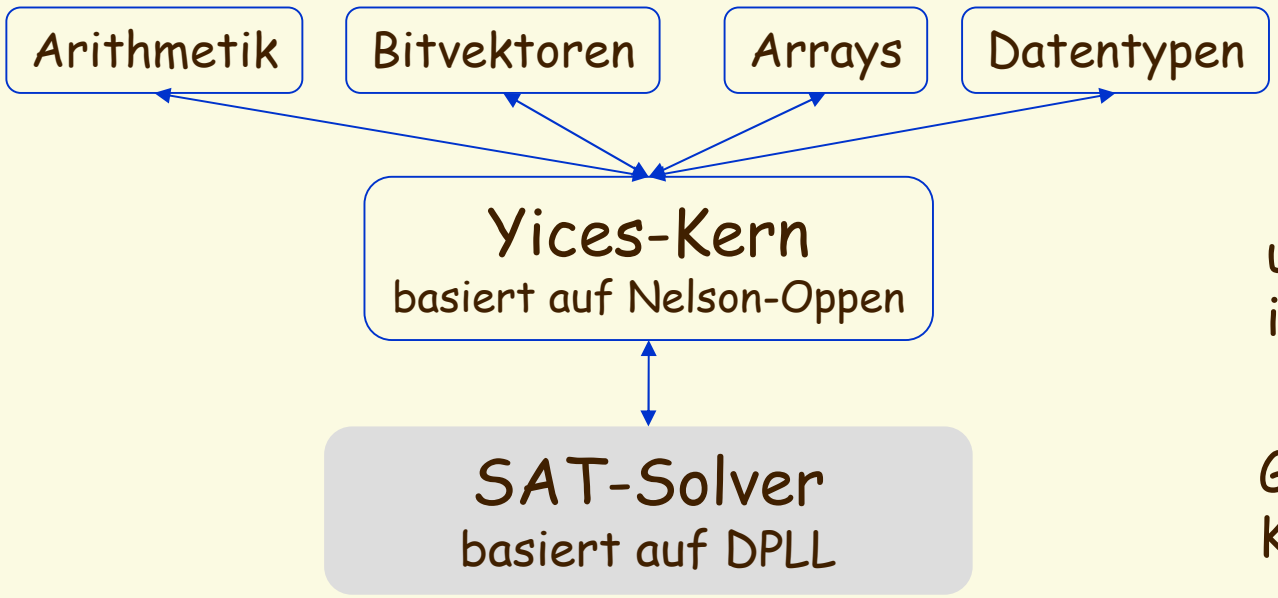
ϕ: $f(x_1) \neq f(x_2) \ \wedge \ g(x_1) \neq g(x_3) \ \wedge \ g(x_2) \neq g(x_3)$

$\mathcal{T}_1$	$\mathcal{T}_2$
$f(x_1) \neq f(x_2)$	$g(x_1) \neq g(x_3)$ $g(x_2) \neq g(x_3)$

Es entstehen keine Gleichungen: NO meldet: "erfüllbar"

SMT-Satisfiability modulo Theories

- Verbindung: SAT-Solver + Proof Checker
- Yices-Architektur:



„Satellite solvers“

u.a.: zerlegt in Theorien

Generiert Kandidaten

SAT modulo Theories - SMT

$$\underbrace{g(a) = c}_{x_1} \wedge \underbrace{(f(g(a)) \neq f(c)}_{\neg x_2} \vee \underbrace{g(a)=d)}_{x_3} \wedge \underbrace{c \neq d}_{\neg x_4}$$

- Sende $\{x_1 , \neg x_2 \vee x_3 , \neg x_4\}$ an SAT-solver.
- SAT-solver liefert erstes Modell: $\{x_1, \neg x_2, \neg x_4\}$.
- Theory solver findet $\{x_1, \neg x_2\}$ E-unerfüllbar.
- Sende $\{x_1 , \neg x_2 \vee x_3 , \neg x_4 , \neg x_1 \vee x_2\}$ an SAT solver.
- SAT solver liefert Modell $\{x_1, x_2, x_3, \neg x_4\}$.
- Theory solver findet $\{x_1, x_2, x_3, \neg x_4\}$ E-unerfüllbar.
- Sende $\{x_1 , \neg x_2 \vee x_3, \neg x_4, \neg x_1 \vee x_2, \neg x_1 \vee \neg x_2 \vee \neg x_3 \vee x_4\}$ an SAT solver.
- •SAT solver berichtet: unerfüllbar

Yices Satisfiability Modulo Theories

- PVS hat ein Reihe von eingebauten Entscheidungsprozeduren
  - sie werden mit (assert) aufgerufen
  - anhand der offenen Formel entscheidet das System,
    - welche Theorien liegen der Formel zugrunde
    - welche Kombinationen von Entscheidungsalgorithmen sind zuständig
    - dann versucht es die Formel zu beweisen
- Die Entscheidungsprozeduren von PVS sind als getrennter Modul Yices verfügbar
  - standalone-modus
  - API für C

Yices ≠ ICS

- Vorgänger ICS mit API für ML
  - open Source
  - http://www.icansolve.com/

Interaktion mit ICS

```
ics> assert f(y - 1) - 1 = y + 1.
:ok s1

ics> show.
:val
v:[v!1 |-> {v!1, v!4}]
u:[v!3 = f(v!2)]
a:[y = -2 + v!3; v!1 = -1 + v!3; v!2 = -3 + v!3]

ics> assert f(x) + 1 = x - 1.
:ok s2
ics> show.
:val
v:[v!1 |-> {v!1, v!4}; v!6 |-> {v!6, v!7}]
u:[v!3 = f(v!2); v!5 = f(x)]
a:[x = 2 + v!5; y = -2 + v!3;
   v!1 = -1 + v!3; v!2 = -3 + v!3;
   v!6 = 1 + v!5]

ics> assert x + 1 = y.
:unsat
```

Interaktion mit Yices

- Webinterface: http://atlantis.seidenberg.pace.edu:8180/yicesweb/index.html

```
(define-type list (datatype (cons car::int cdr::list) nil))
(define P::(-> list bool))
(assert (forall (x::list)
                (implies (and (P x) (cons? x))
                         (and (>= (car x) 0) (P (cdr x))))))
(define x1::list)
(define x2::list)
(define x3::list)
(define x4::list)
(define i1::int)
(define i2::int)
(assert (P x1))
(assert (= x1 (cons i1 x2)))
(assert (= x2 x3))
(assert (= x3 (cons i2 x4)))
(assert (<= i1 1))
(assert (< i2 i1))
(check)
```

Erfüllbar, oder nicht ?

Theorien in Yices

- Uninterpretierte Funktionszeichen: f (x)
- Gleichheit: f (x) = y
- Integer: x < 3 && x >= 2
- Real: 3/5 x + 2y = 17
- Arrays: a[i := v_1][j] = v_2
- Benutzerdefinierte Recursive Datentypen
- (z.B. Listen, Bäume): car (cons(v1, v3)) = v2
- Bitvektoren: concat(bv_1, bv_2) = bv_3

- Beliebige Kombinationen
  - Yices zerlegt die Ausdrücke in Teilausdrücke der einzelnen Theorien + Variablen
  - Nelson-Oppen