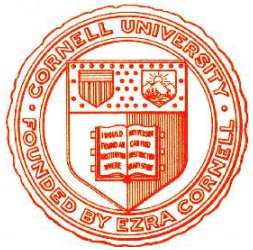
Inferenzmethoden

Einheit 16



Zahlen und Induktion

1. Axiomatische Induktionsbehandlung
2. Induktion mit Theoriekonnektionen
3. Entscheidungsprozedur für Arithmetik

Induktion

Essentiell für mathematische Beweisführung

- **Ermöglicht Schlüsse über unendliche Konzepte**
  - Aussagen über beliebige Zahlen, Listen, Bäume, Graphen, Mengen, ...
  - Eigenschaften von Programmen (unabhängig von der konkreten Eingabe)
- **Grundform: schrittweise Induktion über $\mathbb{N}$**
  - Gilt $P(0)$ und folgt aus $P(x)$ immer $P(x{+}1)$, so gilt P für alle Zahlen
  - Übertragbar auf Listen, Bäume, Strings als strukturelle Induktion
- **Allgemeine Form: strukturelle Induktion**
  - Für Konzepte mit aufwendigerer rekursiver Definition
  - Gilt $P([])$ und folgt $P(a.l)$ aus $P(l)$ für jedes a, so gilt P für alle Listen
  - Gilt $P(\epsilon)$ und folgt $P(wa)$ aus $P(w)$ für jedes a, so gilt P für alle Strings
- **Erweiterung: wohlfundierte Induktion**
  - Reduktion des Problems mit wohlfundierter Ordnung $\succ$
  - Folgt $P(x)$ wenn $P(y)$ für alle $x \succ y$ gilt, so gilt P für alle Elemente
  - Wichtig, wenn Beweisargument "Rückwärtssprünge" macht

Axiomatische Definition natürlicher Zahlen

- **Fest definierte Prädikats- und Funktionssymbole**
  - $\boldsymbol{N(x)}$: x ist eine natürliche Zahl
  - $\boldsymbol{0}$: Konstante Null
  - $\boldsymbol{s(x)}$: Nachfolgerfunktion auf Zahlen ($\equiv x{+}1$)
- **Induktionsaxiome für natürliche Zahlen**

$N(0)$	Erzeugungsaxiom für Null
$\forall x[N(x) \Rightarrow N(s(x))]$	Erzeugungsaxiom für Nachfolger
$\forall x[N(x) \Rightarrow s(x) \dot{\neq} 0]$	Eindeutigkeitsaxiom für Null
$\forall xy[N(x) \wedge N(y) \Rightarrow (s(x) \dot{=} s(y) \Rightarrow x \dot{=} y)]$	Eindeutigkeitsaxiom für Nachfolger
$P[0/x] \wedge \forall y[N(y) \Rightarrow (P[y/x] \Rightarrow P[s(y)/x])]$ $\Rightarrow \forall x(N(x) \Rightarrow P)$	Induktionsschema für jedes Prädikat zu instantiieren

x	*Induktionsvariable*
$P[0/x]$	*Induktionsanfang*
$[N(y) \Rightarrow (P[y/x] \Rightarrow P[s(y)/x])]$	*Induktionsschluß*
$P[y/x]$	*Induktionshypothese*
$P[s(y)/x]$	*Induktionskonklusion*

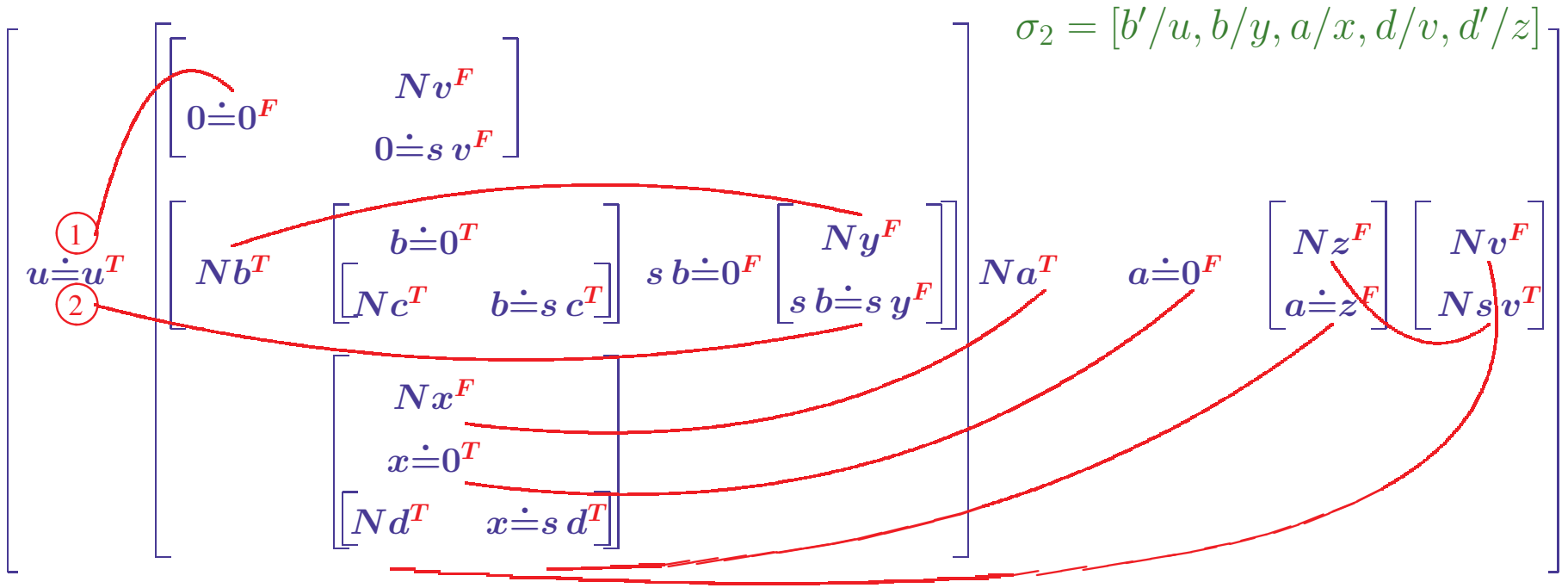
Hinzunahme von Induktionsaxiomen zur Formel

- **Beispiel:** $x \dot{\neq} 0 \Rightarrow \exists z(N(z) \wedge x \dot{=} s(z))$

  Ergänze Gleichheits- und Zahlenaxiome; instantiiere Induktionsschema

  $\forall u\ u \dot{=} u$
  $\wedge\ \forall v\ (N\,v \Rightarrow N(s\,v))$
  $\wedge\ \{[0 \dot{\neq} 0 \Rightarrow \exists v(Nv \wedge 0 \dot{=} s\,v)] \wedge \forall b[Nb \Rightarrow ((b \dot{\neq} 0 \Rightarrow \exists c(Nc \wedge b \dot{=} s\,c)) \Rightarrow (s\,b \dot{\neq} 0 \Rightarrow \exists y(Ny \wedge s\,b \dot{=} s\,y)))]$
  $\Rightarrow \forall x[Nx \Rightarrow (x \dot{\neq} 0 \Rightarrow \exists d(Nd \wedge x \dot{=} s\,d))]\}$
  $\Rightarrow \forall a[Na \Rightarrow (a \dot{\neq} 0 \Rightarrow \exists z(Nz \wedge a \dot{=} s\,z))]$

- **Matrix-Beweis in Nicht-Normalform**



Suchentscheidungen für Induktionsbeweise

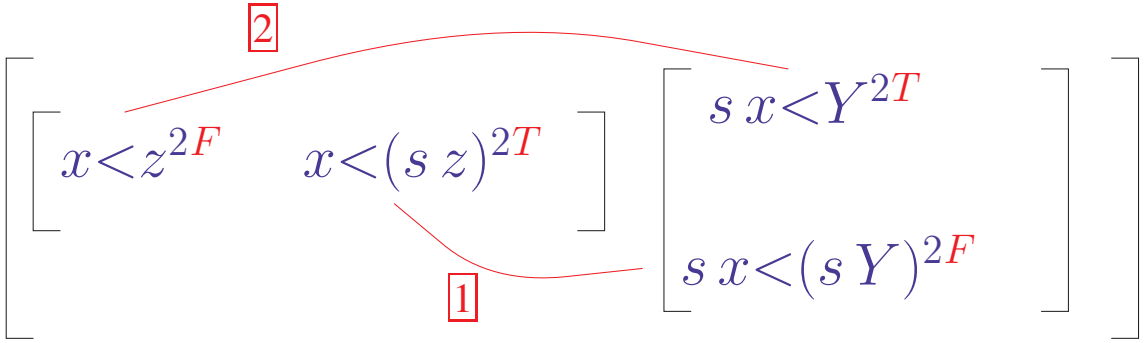
- **Zusätzliche Alternativen bei der Beweisführung**
  1. Ist es nötig, einen Induktionsbeweis zu führen?
  2. Ist eine Verallgemeinerung der zu beweisenden Aussage nötig?
  3. Welche Teilformel ist als Induktionsformel auszuwählen?
  4. Welche Variable der Induktionsformel soll die Induktionsvariable sein ?
  5. Muß eine geschachtelte Induktion durchgeführt werden?
- **Ergibt Suchraum von beträchtlichem Ausmaß**
  - Fragen 1,2 nur vom menschlichem Systembenutzer zu entscheiden
  - Induktionsformel muß engen Zusammenhang zum Beweisziel haben (↦ 3)
  - Anzahl der möglichen Induktionsvariablen (echte Alternativen) ist klein
  - Geschachtelte Induktionen nur wenn weitere Variablen im Induktionsschluß
- **Automatisierung von Induktionsbeweisen ist problematisch**
  - Aber eine stärkere heuristische Steuerung ist möglich
  - Strukturanalyse liefert Menge revelanter (Theorie-)Konnektionen

Heuristische Steuerung von Induktionsbeweisen

Verfeinere Matrixcharakterisierung für Induktionsschritte

- **Induktionsschritt $P[y/x] \Rightarrow P[s\,y/x]$ ist gerichtet** ↦ Einheit 15
  - $P[s\,y/x]$ muß aus $P[y/x]$ arithmetisch folgen
  - Gerichtete Konnektionen mit Theorieimplikationen ersetzen Unifikatoren
- **$P[s\,y/x]$ ist strukturell ähnlich zu $P[y/x]$** ↦ Folie 6
  - Teilformeln von $P[s\,y/x]$ entsprechen denen von $P[y/x]$
  - "Orthogonale" Konnektionen zwischen diesen Teilformeln reichen aus
- **$P[y/x] \Rightarrow P[s\,y/x]$ kann Fallanalyse benötigen** ↦ Folie 7
  - z.B. im Beweis von $\forall x\ \exists y\ (x \geq y^2 \wedge x < (s\,y)^2)$ ist der Induktionsschritt $\exists z(x \geq z^2 \wedge x < (s\,z)^2) \Rightarrow \exists y(s\,x \geq y^2 \wedge s\,x < (s\,y)^2)$ und es muß $s\,x \geq (s\,z)^2$ und $s\,x < (s\,z)^2$ unterschieden werden
  - Erlaube verschiedene (Teil-)Beweise unter verschiedenen Constraints
  - Disjunktion aller Constraints muß allgemeingültig sein
  - Constraints sollten dynamisch erzeugt werden

Erweiterung I: Orthogonale Konnektionen

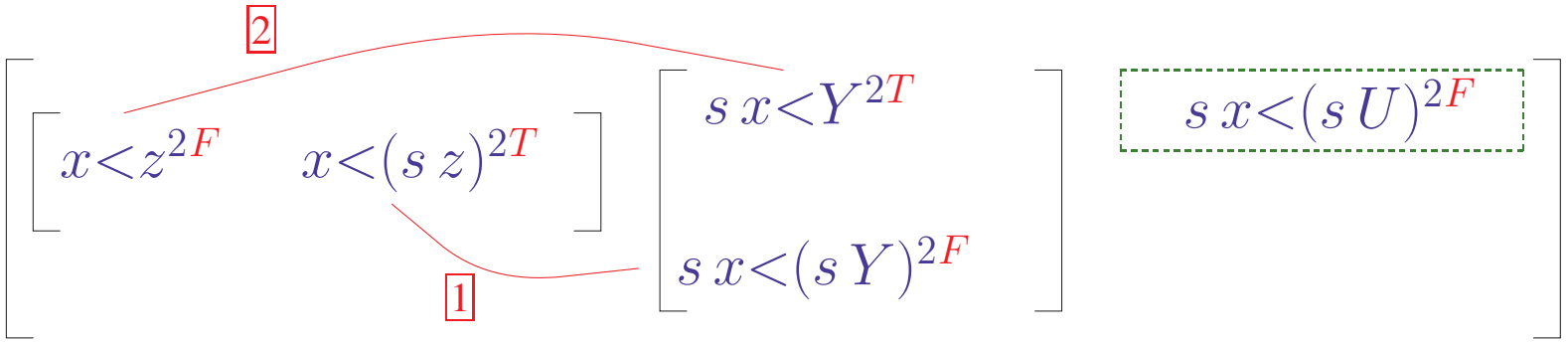


- **(Bezüglich x) Orthogonale Formel $F \equiv H \Rightarrow C$**
  - Formel für die entweder $C = H[\rho(x)/x]$ oder $H = C[\rho(x)/x]$ gilt für eine Substitution ρ (d.h. H und C haben dieselbe Struktur)
- **Orthogonale Konnektion (L^T, L'^F) in $F \equiv H \Rightarrow C$**
  - (L^T, L'^F) ist eine gerichtete Konnektion
  - L hat in H dieselbe relative Position wie L' in C

$\Downarrow$

Eine orthogonale Formel F ist gültig (in $\mathcal{T}$), wenn es eine zulässige Substitution σ gibt, so daß alle orthogonalen Konnektionen in F σ-komplementär sind

Erweiterung II: Constraints



- **Formel F ist σ-komplementär unter Constraint c**
  - Jeder Pfad durch F und c ist σ-komplementär
  - Der Constraint $s\,x{<}(s\,U)^{2F}$ macht den Induktionsschritt gültig
- **$\{c_1, .., c_n\}$ vollständige Menge von Constraints**
  - $\forall x_1..x_k\ c_1 \vee .. \vee c_n$ gültig, wobei $x_1..x_k$ alle freien Variablen der c_i
  - $\{\ s\,x{<}(s\,U)^{2F},\ s\,x{<}(s\,U)^{2T}\}$ wäre vollständig

$\Downarrow$

Eine Formel F ist gültig, wenn es eine vollständige Menge von Constraints $\{c_1,..,c_n\}$ und eine zulässige Substitution σ gibt, so daß F unter jedem Constraint c_i σ-komplementär ist

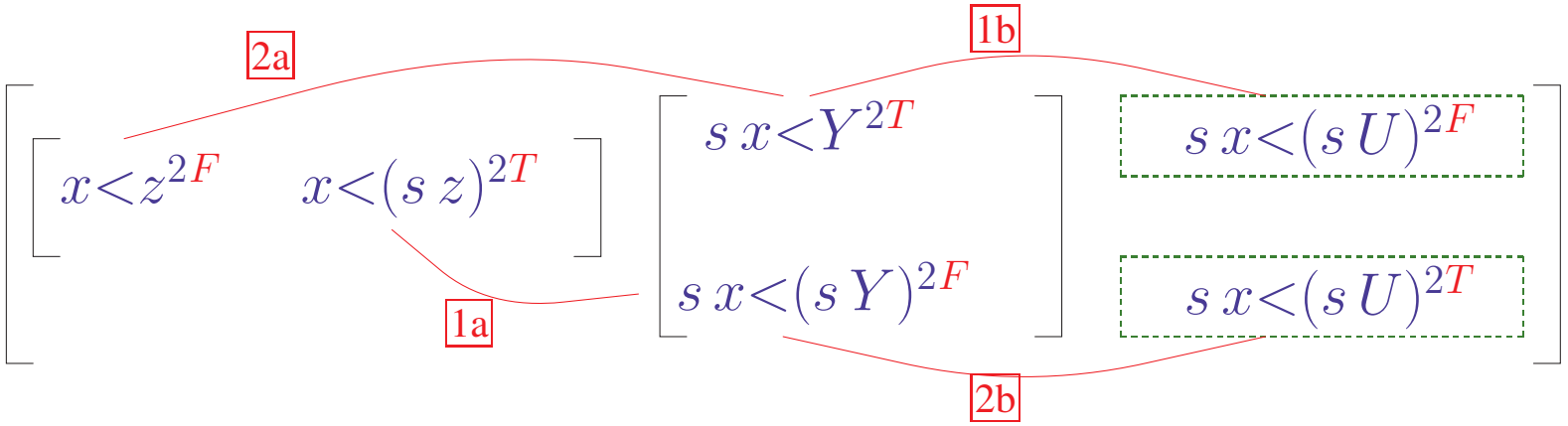
Konstruktion von Constraints

Wenn alle orthogonalen Konnektionen in einer orthogonalen Formel F unter einem atomaren Constraint c^j komplementär sind, dann ist F komplementär unter dem Constraint $(c^1 \wedge .. \wedge c^k)$

- **Konnektion (L^T, L'^F) komplementär unter c^j**
  - (L^T, L'^F) oder (c^j, L'^F) oder (L^T, c^j) ist komplementär
  - Jede Konnektion kann auf diese Art komplementär gemacht werden
- **Constraints liefern iterative Beweismethode**
  - Überprüfe orthogonale Konnektionen
  - Extrahiere atomaren Constraint c^j aus nichtkomplementärer Konnektion
  - F wird komplementär unter $c = (c^1 \wedge .. \wedge c^k)$
  - Prüfe Komplementärität von F unter $\neg c$

  Verfahren erweitert Matrix schrittweise um "tautologische" Klausel
- **Effiziente Integration in lean-Beweiser schwierig**
  - Standard Suchstrategie ist für diesen Schritt nicht verwendbar

Induktionsbeweis für das Integerquadratwurzelproblem



- **Erster Teilbeweis mit orthogonalen Konnektionen**
  - 1a Theorieunifikation mit Rewriting liefert $\sigma_1 = [s\,z/Y]$
  - 1b Zweite Konnektion nicht komplementär $\rightsquigarrow$ Constraint $s\,x<(s\,z)^{2F}$
- **Zweiter Teilbeweis unter Constraint $s\,x<(s\,z)^{2T}$**
  - 2b Konnektion mit Constraint ergibt $\sigma_2 = [z/Y]$ durch Unifikation
  - 2a Instantiierte zweite Konnektion ist komplementär in der Arithmetik
- **Beweis beschreibt implizit einen Algorithmus**

```
sqrt x ≡ falls x=0 dann 0
         sonst setze y = sqrt(x-1)
              falls x<(y+1)^2 dann y sonst y+1
```

Arithmetische Entscheidungsprozeduren

- **Notwendig für praktische Beweisführung**
  - Arithmetisches Schließen taucht fast überall auf
  - Arithmetische Schlüsse liefern keine neuen Erkenntnisse
  - Arithmetische Aussagen tauchen in vielen Erscheinungsformen auf

    `x+1<y, 0<t ⊢ (x+1)*t < y*t` entspricht `x<y, 0<t ⊢ x*t < y*t`

    und `x<y, 0≤t ⊢ x*(t+1) < y*(t+1)` und `x+1≤y, 0<t ⊢ x*t < y*t`
  - Formale Beweise simpler arithmetischer Aussagen sind nicht leicht

    "Wenn drei ganze Zahlen sich jeweils um maximal 1 unterscheiden, dann sind zwei von ihnen gleich"
- **Formale Arithmetik ist unentscheidbar**
  - Theorie ist gleichmächtig mit Theorie der berechenbaren Funktionen
  - Allgemeine Arithmetik ist nicht einmal vollständig axiomatisierbar
- **Entscheidung nur für eingeschränkte Arithmetik**
  - `Arith`: Induktionsfreie Arithmetik
  - `SupInf`: Ganzzahlige lineare Ungleichungssysteme

Arith: Arbeitsweise am Beispiel

$x+2y>z-2,\ i\leq z-5,\ 3x<i+3,\ i\leq 2y+x \Rightarrow 2(y+x)-x \geq 3x+2$

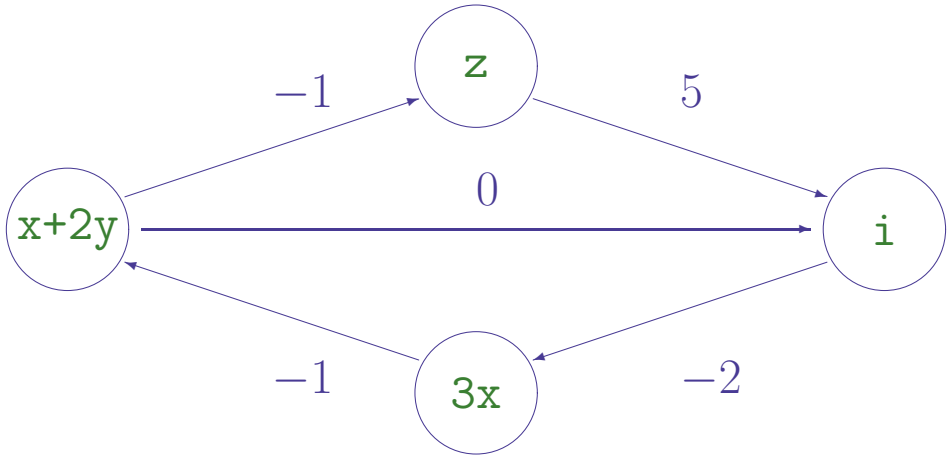
1. Erzeuge Formeln für einen Widerspruchsbeweis:

$\{x+2y>z-2,\ i\leq z-5,\ 3x<i+3,\ i\leq 2y+x,\ 2(y+x)-x < 3x+2\}$

2. Normiere Formeln und erzeuge Ungleichungen der Gestalt $t_1 \geq t_2 \pm j$

$\{x+2y\geq z-1,\ z\geq i+5,\ i\geq 3x-2,\ x+2y\geq i,\ 3x\geq x+2y-1\}$

3. Erzeuge Ordnungsgraphen:



Ordnungsgraph hat positiven Zyklus ... Formel ist gültig

Arith: Theoretisches Fundament

- **Theorie $\mathcal{A}$ der elementar-arithmetischen Aussagen**
  - Quantorenfreie Logik und vordefinierte Symbole $+$, $-$, $*$, $<$ und $=$
  - Alle Variablen sind (implizit) all-quantifiziert
  - Semantik basiert auf Standardaxiomen von $+$, $-$, $*$, $<$ und $=$
  - Keine Induktion, eingeschränkte Substitution
- **$\mathcal{A}$ ist als entscheidbar bekannt**
  - Mathematischer Beweis liefert ineffizientes Entscheidungsverfahren
- **Beweismethode darf klassisch vorgehen**
  - Aussagenlogische Normalisierung der Beweissequenz
  - Normalisierung aller Ungleichungen in $\leq$-Relationen
  - Erzeugung eines Ordnungsgraphen für die $\leq$-Relationen
  - Positive Zyklen im Graphen zeigen daß Sequenz nicht widerlegbar ist

Die formale Theorie $\mathcal{A}$ im Detail

- **Syntax: elementar-arithmetische Formeln** (ohne Induktion!)
  - Terme aufgebaut aus ganzzahligen Konstanten, Variablen und $+$, $-$, $*$
    Andersartige Terme gelten als unspezifierte Konstanten
  - Atomare Formeln: $t_1 \, \rho \, t_2$, wobei t_i Terme, $\rho \in \{<, \leq, >, \geq, =, \neq\}$
  - Formeln aufgebaut aus atomaren Formen mit $\neg$, $\wedge$, $\vee$ und $\Rightarrow$
  - Freie Variablen sind implizit all-quantifiziert
- **Semantik charakterisiert durch Axiome** (Skript §4.3)
  1. Gleichheitsaxiome mit eingeschränkter Substitutivität
  2. Ringaxiome der ganzen Zahlen
  3. Axiome der diskreten linearen Ordnung
  4. Definitionsaxiome für Ordnungsrelationen und Ungleichheiten
  5. Monotonieaxiome
  6. Axiome der Konstantenarithmetik

Grundlagen einer effizienten Entscheidung

- **Widerspruchsbeweise sind in $\mathcal{A}$ auch konstruktiv möglich**
  - *$A_1 \wedge .. A_n \Rightarrow C_1 \vee .. C_m$ gültig gdw. $\{A_1, .. A_n, \neg C_1, .. \neg C_m\}$ widersprüchlich*
- **Konjunktive Normalformen sind auch konstruktiv möglich**
  - *Zu jeder Formel F in $\mathcal{A}$ gibt es eine äquivalente Formel G in KNF*
- **Konstantenfreie Terme sind ersetzbar durch Variablen**
  - *Eine Menge von instantiierten $\mathcal{A}$-Formeln $F_i[e_1..e_k/u_1..u_k]$ ist genau dann widersprüchlich, wenn $\{F_1, .. F_n\}$ widersprüchlich ist (e_i konstantenfrei)*

    $\Leftarrow$: trivial, $\Rightarrow$: Der Widerspruchsbeweis läßt sich (mühsam) übertragen
- **Ordnungsgraphen codieren lineare Ungleichungen**
  - *$\Gamma = v_1 \geq u_1 + c_1, \ldots, v_n \geq u_n + c_n$ ist genau dann widersprüchlich, wenn der Ordnungsgraph $\mathcal{G}$ von Γ einen positiven Zyklus besitzt.*
  - Der Ordnungsgraph von Γ besteht aus den Knoten $\{u_1, .. u_n, v_1, .. v_n\}$ und den Kanten $\{v_1 \xrightarrow{c_1} u_1, .., v_n \xrightarrow{c_n} u_n\}$. Ein positiver Zyklus ist eine Serie von Kanten $[k_1 \xrightarrow{g_1} k_2,\ k_2 \xrightarrow{g_2} k_3, .., \ k_m \xrightarrow{g_m} k_1]$ mit Gewicht $\sum_{i=1}^{m} g_i > 0$.

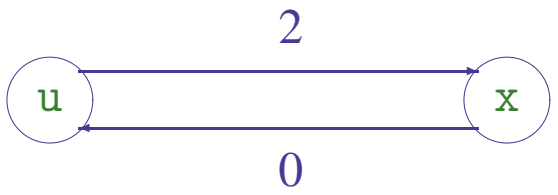
Arith: Verarbeitung einer Formel $A_1 \wedge .. \wedge A_n \Rightarrow C_1 \vee .. \vee C_m$

1. **Transformiere Sequenz in Liste atomarer arithmetischer Formeln**
   - Umwandlung in Widerspruchsbeweis für $\{A_1, ..A_n, \neg C_1, .. \neg C_m\}$
   - Zerlege Konjunktionen in den C_i in atomare Formeln
   - Entferne Formeln, die keine arithmetischen Ungleichungen sind
   - Ersetze nichtarithmetische Ausdrücke in Termen durch Variablen
2. **Eliminiere Ungleichungen der Form $x \neq y$**
   - Transformiere $x \neq y$ in die (nichtatomare) Formel $x \geq y+1 \vee y \geq x+1$
   - Erzeuge DNF und betrachte jede Konjunktion separat
3. **Transformiere Terme in monadische lineare Polynome $u_i + c_i$**
   - Normalisiere Komparanden jeder Ungleichung zu Standardpolynomen
   - Ersetze nicht-konstante Anteile der Polynome durch neue Variablen u_i
4. **Konvertiere jede Formel in eine Ungleichung der Gestalt $t_1 \geq t_2$**
   t_1 Variable oder 0; t_2 monadisches lineares Polynom
5. **Suche positive Zyklen im Ordnungsgraphen der Formelmenge**
   - Im Erfolgsfall generiere Wohlgeformtheitsziele für alle "Variablen"

Arith Arbeitsweise: Beispiel 1

Beweise $x+1<y^2 \Rightarrow x<y^2$

1. **Erzeuge Formel für Widerspruchsbeweis:** $x+1<y^2,\ \neg(x<y^2)$
   Nach Auflösung der Negation $x+1<y^2,\ x\geq y^2$
2. **Entferne Literale ohne atomare arithmetische Formeln** ✓
3. **Transformiere Ungleichungen $x\neq y$ in $x\geq y+1 \vee y\geq x+1$** ✓
4. **Transformiere Terme in monadische lineare Polynome**
   $x+1<u,\ x\geq u$
5. **Konvertiere in Ungleichungen der Gestalt $u_i\geq c+u_j$**
   $u\geq 2+x,\ x\geq 0+u$
6. **Erzeuge den Ordnungsgraphen der Klausel**



7. **Standardalgorithmus findet positiven Zyklus im Graphen**
   Ausgangsformel war gültig

Arith Arbeitsweise: Beispiel 2

Beweise $z-1<(x+y)^2 \wedge (x+y)^2<z+1 \Rightarrow z=(x+y)^2$

1. Erzeuge Beweisklausel: $z-1<(x+y)^2,\ (x+y)^2<z+1,\ z\neq(x+y)^2$

2. Entferne Literale ohne atomare arithmetische Formeln ✓

3. Transformiere Ungleichungen $x\neq y$ in $x\geq y+1 \vee y\geq x+1$

1. $z-1<(x+y)^2,\ (x+y)^2<z+1,\ z<(x+y)^2$
2. $z-1<(x+y)^2,\ (x+y)^2<z+1,\ z>(x+y)^2$

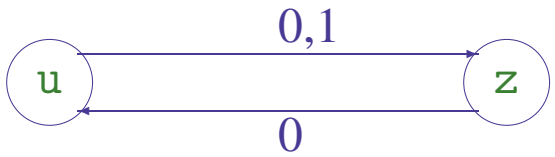
4. Transformiere Terme in monadische lineare Polynome

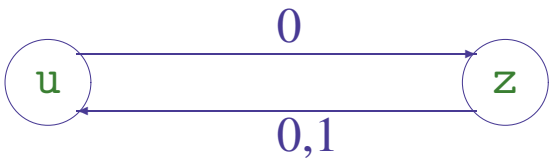
1. $z-1<u,\ u<z+1,\ z<u$ 2. $z-1<u,\ u<z+1,\ z>u$

5. Konvertiere in Ungleichungen der Gestalt $u_i\geq c+u_j$

1. $u\geq 0+z,\ z\geq 0+u,\ u\geq 1+z$ 2. $u\geq 0+z,\ z\geq 0+u,\ z\geq 1+u$

6. Erzeuge die Ordnungsgraphen der Klauseln





7. Standardalgorithmus findet je einen positiven Zyklus

Ausgangsformel war gültig

Arith: Stärken und Schwächen

- **Effizient ausführbar**
  - Es gibt Standardalgorithmen für Zyklensuche in gewichteten Graphen
  - Exponentielle worst-case Komplexität (in $\neq$) praktisch unbedeutend
- **Beschränkt auf triviale Monotonieschlüsse**
  - Normierung der Terme enthält Anwendung des Monotonieaxioms
    $\texttt{x} \geq \texttt{y} \wedge \texttt{i} \geq \texttt{j} \Rightarrow \texttt{x+i} \geq \texttt{y+j}$ mit Integerkonstanten `i` und `j`
  - Nichtriviale Monotonien müssen separat behandelt werden
    (Monotonie von +/− mit Variablen oder Monotonien mit $*$)
- **Zu schwach für lineare Ungleichungssysteme**
  - Monadische Polynome zerstören Bezüge zwischen Ungleichungen

Andere Arten der Induktionsbehandlung

- **Verwendung von wohlfundierter Induktion**
  - $P[0/x] \wedge \forall y[N(y) \Rightarrow (P[y/x] \Rightarrow P[s\,y/x])] \Rightarrow \forall x(N(x) \Rightarrow P)$
  - Standardinduktion führt zu einfach strukturierter Beweisführung
  - $\forall x(N(x) \Rightarrow \forall y[N(y) \Rightarrow (x \succ y \Rightarrow P[y/x])] \Rightarrow F) \Rightarrow F$
  - Vollständige Induktion liefert elegantere Beweise, gleiche Beweisstärke
  - Ordnung $\succ$ muß wohlfundiert sein
- **Konnektionsschemata für Induktion**
  - Das Extensionsverfahren mit Axiomen ist nicht vollständig

    Für Induktionsbeweise gilt kein Schnitteliminationssatz
  - Gegenstück zur Induktionsregel des Sequenzenkalküls erforderlich
- **Induktionslose Induktion mit Rewriting** ↦ Einheit 17
  - Superposition von Regelsystemen für arithmetische Gleichungen
- **Definition von Zahlen in Logik zweiter Stufe** ↦ Einheit 14
  - Kein Induktionsschema erforderlich
  - Eleganter und vollständig, aber schwerer zu automatisieren

ANHANG

Die Entscheidungsprozedur SupInf

- **Entscheide lineare Ungleichungen über $\mathbb{Z}$**
  - Sinnvoll in Anwendungen für die `Arith` zu schwach ist
- **Anpassung von Bledsoe's Sup-Inf Methode** (1975)
  - Extrahiere aus Sequenz eine Menge linearer Ungleichungen $0 \leq e_i$, deren Unerfüllbarkeit die Gültigkeit der Sequenz impliziert
  - Bestimme obere und untere Grenzen für die Variablen der e_i
  - Wenn alle Variablen in $\mathbb{Z}$ erfüllbar sind, liefere Gegenbeispiel
- **Logische Theorie: Arithmetische Formeln**
  - Kombination von Ungleichungen über arithmetischen Typen
- **Korrekt aber unvollständig über $\mathbb{Z}$**
  - Bledsoe's Methode ist nur für rationale Zahlen korrekt und vollständig
  - `SupInf` ist dennoch hilfreich in der Praxis

Die Sup-Inf Basismethode

Analysiere lineare Ungleichungsmengen über $\mathbb{Q}$

- **Betrachte Formeln der Form** $0 \leq e_1 \wedge .. \wedge 0 \leq e_n$
  - e_i lineare Ausdrücke über rationalen Variablen $x_1, .., x_m$
  - Suche Belegung der x_j, welche die Konjunktion erfüllen
- **Bestimme obere/untere Grenzen für Werte der** x_j
  - Aufwendiges Verfahren(!) verbessert obere und untere Schranken iterativ
  - Resultierende Schranken sind nachweislich optimal (Shostak 1977)
    Methode liefert Suprema und Infima für Belegungen der x_j
  - Erfüllende Belegung existiert g.d.w. Infima jeweils kleiner als Suprema
- **Keine "echte" Entscheidung über** $\mathbb{Z}$
  - Korrekt: Unerfüllbarkeit über $\mathbb{Q}$ bedeutet Unerfüllbarkeit über $\mathbb{Z}$
  - Unvollständig: Erfüllende Belegung über $\mathbb{Q}$ liefert evtl. keine über $\mathbb{Z}$
    Reparatur möglich mit Integer Linear Programming ($\mathcal{NP}$-vollständig)

SupInf: Arbeitsweise am Beispiel

1. Anfangssequenz: `x+z<5, 3*z≥y+8, x<y ⊢ 2+z>2*y ∨ x>z-5`

2. Extrahiere arithmetische Formel für Widerspruchsbeweis

`x+z<5 ∧ 3*z≥y+8 ∧ x<y ∧ ¬(2+z>2*y) ∧ ¬(x>z-5) ⊢ ff`

3. Transformiere Formel in disjunktive Normalform über $\leq$

`x+z+1≤5 ∧ y+8≤3*z ∧ x+1≤y ∧ 2+z≤2*y ∧ x≤z-5`

4. Normalisiere Ungleichungen in die Form $0 \leq p_i$

`0≤4-x-z ∧ 0≤3*z-y-8 ∧ 0≤y-x-1 ∧ 0≤2*y-z-2 ∧ 0≤z-x-5`

5. Wende iterativ die Sup-Inf Basismethode an

`SUP(x)=-3  INF(x)=-∞,  SUP(y)=1  INF(y)=14/5`

6. `SUP(y)<INF(y)...`

– Arithmetische Formel kann nicht erfüllt werden **Sequenz ist gültig**

SupInf: Widerlegungsbeispiel

1. Anfangssequenz: $x<3*y+2,\ x=1 \vdash x=y$

2. Extrahiere arithmetische Formel für Widerspruchsbeweis

$x<3*y+2 \ \wedge\ x=1 \ \wedge\ x\neq y \vdash \mathit{ff}$

3. Setze Gleichheiten in andere Ungleichungen ein

$1<3*y+2 \ \wedge\ 1\neq y \vdash \mathit{ff}$

4. Transformiere Formel in disjunktive Normalform über $\leq$

$(2\leq 3*y+2 \ \wedge\ 2\leq y) \ \vee\ (2\leq 3*y+2 \ \wedge\ y\leq 0)$

5. Normalisiere Ungleichungen in die Form $0\leq p_i$

$(0\leq 3*y \ \wedge\ 0\leq y-2) \ \vee\ (0\leq 3*y \ \wedge\ 0\leq -y)$

6. WendeSup-Inf Basismethode auf jedes Disjunkt an

1. $\mathit{SUP}(y)=\infty \quad \mathit{INF}(y)=2$ **2.** $\mathit{SUP}(y)=0 \quad \mathit{INF}(y)=0$

7. $\{z:\mathbb{Z} \mid \mathit{SUP}(y)\geq z\geq \mathit{INF}(y)\}$ **ist nicht leer ...**

– Es gibt ein ganzzahliges Gegenbeispiel **Sequenz ist ungültig**

SupInf: Formale Grundkonzepte

- **Arithmetische Typen**
  - $\mathbb{Z}$ (`int`), $\mathbb{Z}^{-0}$ (`int_nzero`)
  - $\mathbb{N}$ (`nat`), $\mathbb{N}^{+}$ (`nat_plus`)
  - $\{i\,...\}$ (`int_upper`)
  - $\{i\,...\,j^{-}\}$ (`int_seg`), $\{i\,...\,j\}$ (`int_iseg`)
- **Arithmetische Literale**
  - $a{=}b \in T$ oder $a{\neq}b \in T$, wobei T arithmetischer Typ
  - Arithmetische Ungleichungen mit $<$, $\leq$, $>$ und $\geq$
  - Negationen arithmetischer Literale
- **Arithmetische Formeln**
  - (Verschachtelte) Konjunktionen und Disjunktionen arithmetischer Literale

SupInf: Allgemeine Arbeitsweise

Anfangssequenz: $\Gamma, r_1, \ldots, r_n \vdash r_0$ (r_i arithmetische Formel)

1. **Extrahiere arithmetische Formel** $F = r_1 \wedge \ldots \wedge r_n \wedge \neg r_0$
   – Aus Unerfüllbarkeit von F folgt Gültigkeit der Anfangssequenz
2. **Transformiere F in disjunktive Normalform über $\leq$**
   – $x<y$ bzw. $y>x$ wird umgewandelt in $x+1\leq y$,
   – $x\neq y$ wird $x+1\leq y \vee y+1\leq x$
   – $x=y$ wird, wenn möglich, durch Substitution aufgelöst
3. **Normalisiere Ungleichungen in die Form** $0\leq p_i$
   – p_i sind Standardrepräsentationen von Polynomen
4. **Ersetze nichtlineare Teilausdrücke durch Variablen**
5. **Wende Sup-Inf Basismethode auf jedes Disjunkt an**
   – Wenn jedes Disjunkt unerfüllbar ist, erzeuge Wohlgeformtheitsziele
   – Andernfalls liefert `supinf_info` erfüllende Belegung als Gegenbeispiel