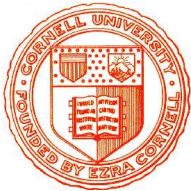


Automatisierte Logik und Programmierung



Lektion 14

Entscheidungsprozeduren



1. Einsatzbereiche
2. **Arith**: elementare Arithmetik
3. **SupInf**: Lineare Ungleichungen über $\mathbb{Z}$
4. **Eq**: Typfreie Gleichheiten
5. Grenzen der Anwendbarkeit

- **Sinnvoll** für “uninteressante” Beweisziele
 - Problem ist Variation bekannter mathematischer Erkenntnisse
 - + Beweisdetails / Extraktterm ohne Bedeutung (nur Wahrheit gefragt)
 - + Formaler Beweis mit Taktiken zu aufwendig

- **Sinnvoll** für “uninteressante” Beweisziele
 - Problem ist Variation bekannter mathematischer Erkenntnisse
 - + Beweisdetails / Extraktterm ohne Bedeutung (nur Wahrheit gefragt)
 - + Formaler Beweis mit Taktiken zu aufwendig
- **Möglich** für algorithmisch entscheidbare Ziele
 - Schneller Test, ob Verfahren überhaupt anwendbar ist
 - + Maschinennahe Charakterisierung für Gültigkeit vorhanden
 - + Effizientes Entscheidungsverfahren programmierbar

- **Sinnvoll** für “uninteressante” Beweisziele
 - Problem ist Variation bekannter mathematischer Erkenntnisse
 - + Beweisdetails / Extraktterm ohne Bedeutung (nur Wahrheit gefragt)
 - + Formaler Beweis mit Taktiken zu aufwendig
- **Möglich** für algorithmisch entscheidbare Ziele
 - Schneller Test, ob Verfahren überhaupt anwendbar ist
 - + Maschinennahe Charakterisierung für Gültigkeit vorhanden
 - + Effizientes Entscheidungsverfahren programmierbar
- **Erforderlich:** externe Verifikation der Prozedur
 - Korrektheit und Vollständigkeit der Entscheidungsalgorithmen
 - Konsistenz mit dem Rest der Theorie (Typkonzept!)

- **Sinnvoll** für “uninteressante” Beweisziele
 - Problem ist Variation bekannter mathematischer Erkenntnisse
 - + Beweisdetails / Extraktterm ohne Bedeutung (nur Wahrheit gefragt)
 - + Formaler Beweis mit Taktiken zu aufwendig
- **Möglich** für algorithmisch entscheidbare Ziele
 - Schneller Test, ob Verfahren überhaupt anwendbar ist
 - + Maschinennahe Charakterisierung für Gültigkeit vorhanden
 - + Effizientes Entscheidungsverfahren programmierbar
- **Erforderlich:** externe Verifikation der Prozedur
 - Korrektheit und Vollständigkeit der Entscheidungsalgorithmen
 - Konsistenz mit dem Rest der Theorie (Typkonzept!)
 - In Nuprl bisher nur für Arithmetik und Gleichheit
 - Prozeduren für Listen, Kongruenzabschluß etc. noch nicht integriert

- **Notwendig** für praktische Beweisführung

- **Notwendig** für praktische Beweisführung
 - Arithmetisches Schließen taucht fast überall auf

- **Notwendig** für praktische Beweisführung
 - Arithmetisches Schließen taucht **fast überall** auf
 - Arithmetische Schlüsse liefern **keine neuen Erkenntnisse**

- **Notwendig** für praktische Beweisführung

- Arithmetisches Schließen taucht fast überall auf
- Arithmetische Schlüsse liefern keine neuen Erkenntnisse
- Arithmetische Aussagen tauschen in vielen Erscheinungsformen auf

$x+1 < y, 0 < t \vdash (x+1) * t < y * t$ entspricht $x < y, 0 < t \vdash x * t < y * t$
und $x < y, 0 \leq t \vdash x * (t+1) < y * (t+1)$ und $x+1 \leq y, 0 < t \vdash x * t < y * t$

- **Notwendig** für praktische Beweisführung

- Arithmetisches Schließen taucht fast überall auf
- Arithmetische Schlüsse liefern keine neuen Erkenntnisse
- Arithmetische Aussagen tauschen in vielen Erscheinungsformen auf
 $x+1 < y, 0 < t \vdash (x+1) * t < y * t$ entspricht $x < y, 0 < t \vdash x * t < y * t$
und $x < y, 0 \leq t \vdash x * (t+1) < y * (t+1)$ und $x+1 \leq y, 0 < t \vdash x * t < y * t$
- Formale Beweise simpler arithmetischer Aussagen sind nicht leicht
*Wenn drei ganze Zahlen sich jeweils um maximal 1 unterscheiden,
dann sind zwei von ihnen gleich*

- **Notwendig** für praktische Beweisführung

- Arithmetisches Schließen taucht **fast überall** auf
- Arithmetische Schlüsse liefern **keine neuen Erkenntnisse**
- Arithmetische Aussagen tauschen **in vielen Erscheinungsformen** auf
$$x+1 < y, 0 < t \vdash (x+1) * t < y * t \text{ entspricht } x < y, 0 < t \vdash x * t < y * t$$

und $x < y, 0 \leq t \vdash x * (t+1) < y * (t+1)$ und $x+1 \leq y, 0 < t \vdash x * t < y * t$
- **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

- **Notwendig** für praktische Beweisführung

- Arithmetisches Schließen taucht fast überall auf
- Arithmetische Schlüsse liefern keine neuen Erkenntnisse
- Arithmetische Aussagen tauschen in vielen Erscheinungsformen auf
 $x+1 < y, 0 < t \vdash (x+1)*t < y*t$ entspricht $x < y, 0 < t \vdash x*t < y*t$
und $x < y, 0 \leq t \vdash x*(t+1) < y*(t+1)$ und $x+1 \leq y, 0 < t \vdash x*t < y*t$
- 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

- **Notwendig** für praktische Beweisführung

- Arithmetisches Schließen taucht **fast überall** auf
- Arithmetische Schlüsse liefern **keine neuen Erkenntnisse**
- Arithmetische Aussagen tauschen **in vielen Erscheinungsformen** auf
$$x+1 < y, 0 < t \vdash (x+1) * t < y * t \text{ entspricht } x < y, 0 < t \vdash x * t < y * t$$

und $x < y, 0 \leq t \vdash x * (t+1) < y * (t+1)$ und $x+1 \leq y, 0 < t \vdash x * t < y * t$
- **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**

- **Notwendig** für praktische Beweisführung

- Arithmetisches Schließen taucht fast überall auf
- Arithmetische Schlüsse liefern keine neuen Erkenntnisse
- Arithmetische Aussagen tauschen in vielen Erscheinungsformen auf
$$x+1 < y, 0 < t \vdash (x+1)*t < y*t \text{ entspricht } x < y, 0 < t \vdash x*t < y*t$$
und $x < y, 0 \leq t \vdash x*(t+1) < y*(t+1)$ und $x+1 \leq y, 0 < t \vdash x*t < y*t$
- 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

Entscheide Probleme der induktionsfreien Arithmetik

Entscheide Probleme der induktionsfreien Arithmetik

- **Anfangssequenz:** $\Gamma \vdash C_1 \vee \dots \vee C_m$
 - C_i arithmetische Relationen über $\mathbb{Z}$

Entscheide Probleme der induktionsfreien Arithmetik

- **Anfangssequenz:** $\Gamma \vdash C_1 \vee \dots \vee C_m$
 - C_i arithmetische Relationen über $\mathbb{Z}$
- **Logische Theorie: quantorenfreie Arithmetik**
 - Aussagenlogische Kombination von Ungleichungen über einfachen arithmetischen Termen
 - Standardaxiome von $+$, $-$, $*$, $<$ und $=$

Entscheide Probleme der induktionsfreien Arithmetik

- **Anfangssequenz:** $\Gamma \vdash C_1 \vee \dots \vee C_m$
 - C_i arithmetische Relationen über $\mathbb{Z}$
- **Logische Theorie: quantorenfreie Arithmetik**
 - Aussagenlogische Kombination von Ungleichungen über einfachen arithmetischen Termen
 - Standardaxiome von $+$, $-$, $*$, $<$ und $=$
- **Beweismethode:**
 - Transformiere Sequenz in gerichteten Graph mit gewichteten Kanten

Entscheide Probleme der induktionsfreien Arithmetik

- **Anfangssequenz:** $\Gamma \vdash C_1 \vee \dots \vee C_m$
 - C_i arithmetische Relationen über $\mathbb{Z}$
- **Logische Theorie: quantorenfreie Arithmetik**
 - Aussagenlogische Kombination von Ungleichungen über einfachen arithmetischen Termen
 - Standardaxiome von $+$, $-$, $*$, $<$ und $=$
- **Beweismethode:**
 - Transformiere Sequenz in gerichteten Graph mit gewichteten Kanten
 - Teste ob positive Zyklen im Graph vorkommen

Entscheide Probleme der induktionsfreien Arithmetik

- **Anfangssequenz:** $\Gamma \vdash C_1 \vee \dots \vee C_m$
 - C_i arithmetische Relationen über $\mathbb{Z}$
- **Logische Theorie: quantorenfreie Arithmetik**
 - Aussagenlogische Kombination von Ungleichungen über einfachen arithmetischen Termen
 - Standardaxiome von $+$, $-$, $*$, $<$ und $=$
- **Beweismethode:**
 - Transformiere Sequenz in gerichteten Graph mit gewichteten Kanten
 - Teste ob positive Zyklen im Graph vorkommen
 - Implementierung in Nuprl als Lisp Prozedur

Entscheide Probleme der induktionsfreien Arithmetik

- **Anfangssequenz:** $\Gamma \vdash C_1 \vee \dots \vee C_m$
 - C_i arithmetische Relationen über $\mathbb{Z}$
- **Logische Theorie: quantorenfreie Arithmetik**
 - Aussagenlogische Kombination von Ungleichungen über einfachen arithmetischen Termen
 - Standardaxiome von $+$, $-$, $*$, $<$ und $=$
- **Beweismethode:**
 - Transformiere Sequenz in gerichteten Graph mit gewichteten Kanten
 - Teste ob positive Zyklen im Graph vorkommen
 - Implementierung in Nuprl als Lisp Prozedur
- **Eingebettet in die Taktik Auto**

Syntax: *elementar-arithmetische Formeln*

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet
- **Atomare Formeln**: $t_1 \rho t_2$, wobei t_i Terme, $\rho \in \{<, \leq, >, \geq, =, \neq\}$

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet
- **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 $\Rightarrow$

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet
- **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 $\Rightarrow$
- Alle freien **Variablen** sind implizit all-quantifiziert
- Keine induktive Konstruktion

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet
- **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 $\Rightarrow$
- Alle freien **Variablen** sind implizit all-quantifiziert
- Keine induktive Konstruktion

Semantik charakterisiert durch Axiome

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet
- **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 $\Rightarrow$
- Alle freien **Variablen** sind implizit all-quantifiziert
- Keine induktive Konstruktion

Semantik charakterisiert durch Axiome

- Gleichheitsaxiome mit eingeschränkter Substitutivität

$$x=x$$

(Reflexivität)

$$x=y \Rightarrow y=x$$

(Symmetrie)

$$x=y \wedge y=z \Rightarrow x=z$$

(Transitivität)

$$x=y \wedge x \rho z \Rightarrow y \rho z$$

(eingeschränkte Substitutivität)

$$x=y \wedge z \rho x \Rightarrow z \rho y$$

Aus $x=z \wedge x \neq x*y$ folgt $z \neq x*y$, aber nicht $z \neq z*y$

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet
- **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 $\Rightarrow$
- Alle freien **Variablen** sind implizit all-quantifiziert
- Keine induktive Konstruktion

Semantik charakterisiert durch Axiome

- **Gleichheits**saxiome mit eingeschränkter Substitutivität
- Axiome der **Konstantenarithmetik**
 $1+1=2, 2+1=3, 3+1=4, \dots$

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet
- **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 $\Rightarrow$
- Alle freien **Variablen** sind implizit all-quantifiziert
- Keine induktive Konstruktion

Semantik charakterisiert durch Axiome

- **Gleichheitsaxiome** mit eingeschränkter Substitutivität
- Axiome der **Konstantenarithmetik**
- **Ringaxiome** der ganzen Zahlen

$$x+y=y+x \quad x*y=y*x$$

(Kommutativgesetze)

$$(x+y)+z=x+(y+z) \quad (x*y)*z=x*(y*z)$$

(Assoziativgesetze)

$$x*(y+z)=(x*z)+(y*z)$$

(Distributivgesetz)

$$x+0=x \quad x*1=x$$

(Neutrale Elemente)

$$x+(-x)=0$$

(Inverses Element der Addition)

$$x-y=x+(-y)$$

(Definition der Subtraktion)

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet
- **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 $\Rightarrow$
- Alle freien **Variablen** sind implizit all-quantifiziert
- Keine induktive Konstruktion

Semantik charakterisiert durch Axiome

- **Gleichheits**axiome mit eingeschränkter Substitutivität
- Axiome der **Konstantenarithmetik**
- **Ringaxiome** der ganzen Zahlen
- Axiome der **diskreten linearen Ordnung**
 - $\neg(x < x)$ (Irreflexivität)
 - $x < y \vee x = y \vee y < x$ (Trichotomie)
 - $x < y \wedge y < z \Rightarrow x < z$ (Transitivität)
 - $\neg(x < y \wedge y < x+1)$ (Diskretheit)

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet
- **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 $\Rightarrow$
- Alle freien **Variablen** sind implizit all-quantifiziert
- Keine induktive Konstruktion

Semantik charakterisiert durch Axiome

- **Gleichheitsaxiome** mit eingeschränkter Substitutivität
- Axiome der **Konstantenarithmetik**
- **Ringaxiome** der ganzen Zahlen
- Axiome der **diskreten linearen Ordnung**
- **Definitionsaxiome** für Ordnungsrelationen und Ungleichheiten

$$x \neq y \Leftrightarrow \neg(x=y)$$

$$x > y \Leftrightarrow y < x$$

$$x \leq y \Leftrightarrow x < y \vee x = y$$

$$x \geq y \Leftrightarrow y < x \vee x = y$$

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet
- **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 $\Rightarrow$
- Alle freien **Variablen** sind implizit all-quantifiziert
- Keine induktive Konstruktion

Semantik charakterisiert durch Axiome

- **Gleichheitsaxiome** mit eingeschränkter Substitutivität
- Axiome der **Konstantenarithmetik**
- **Ringaxiome** der ganzen Zahlen
- Axiome der **diskreten linearen Ordnung**
- **Definitionsaxiome** für Ordnungsrelationen und Ungleichheiten
- **Monotonieaxiome**

$$x \geq y \wedge z \geq w \Rightarrow x+z \geq y+w$$

(Addition)

$$x \geq y \wedge z \leq w \Rightarrow x-z \geq y-w$$

(Subtraktion)

$$x \geq 0 \wedge y \geq z \Rightarrow x*y \geq x*z$$

(Multiplikation)

$$x > 0 \wedge x*y \geq x*z \Rightarrow y \geq z$$

(Faktorisierung)

Syntax: **elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme werden als Konstanten betrachtet
- **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 $\Rightarrow$
- Alle freien **Variablen** sind implizit all-quantifiziert
- Keine induktive Konstruktion

Semantik charakterisiert durch Axiome

- **Gleichheitsaxiome** mit eingeschränkter Substitutivität
- Axiome der **Konstantenarithmetik**
- **Ringaxiome** der ganzen Zahlen
- Axiome der **diskreten linearen Ordnung**
- **Definitionsaxiome** für Ordnungsrelationen und Ungleichheiten
- **Monotonieaxiome**

$\mathcal{A}$ ist als entscheidbar bekannt

- Beweis liefert nur ein ineffizientes Verfahren

● Abtrennung der Monotonie

- **Triviale Monotonien**: Monotonieaxiome von $+$ und $-$ mit Konstanten
- **monotonicity** Regel für Anwendung nichttrivialer Monotonien
 - verwendet Monotonietabellen für Kombination von Ungleichungen
 - erzeugt zusätzliche (kombinierte) Hypothesen

- **Abtrennung der Monotonie**

- **Triviale Monotonien**: Monotonieaxiome von $+$ und $-$ mit Konstanten
- **monotonicity** Regel für Anwendung nichttrivialer Monotonien
 - verwendet Monotonietabellen für Kombination von Ungleichungen
 - erzeugt zusätzliche (kombinierte) Hypothesen

- **Erzeugung der Konjunktiven Normalform**

- Zu jeder Formel F in $\mathcal{A}$ gibt es in $\mathcal{A}$ eine äquivalente Formel G in KNF
- Man kann jedes Konjunkt isoliert beweisen

● Abtrennung der Monotonie

- **Triviale Monotonien**: Monotonieaxiome von $+$ und $-$ mit Konstanten
- **monotonicity** Regel für Anwendung nichttrivialer Monotonien
 - verwendet Monotonietabellen für Kombination von Ungleichungen
 - erzeugt zusätzliche (kombinierte) Hypothesen

● Erzeugung der **Konjunktiven Normalform**

- Zu jeder Formel F in $\mathcal{A}$ gibt es in $\mathcal{A}$ eine äquivalente Formel G in KNF
- Man kann jedes Konjunkt isoliert beweisen

● **Widerspruchsbeweis**

- $\Gamma \vdash C_1 \vee \dots \vee C_n$ gültig, g.d.w. $\Gamma, \neg C_1, \dots, \neg C_n \vdash \text{ff}$ gültig

● Abtrennung der Monotonie

- **Triviale Monotonien**: Monotonieaxiome von $+$ und $-$ mit Konstanten
- **monotonicity** Regel für Anwendung nichttrivialer Monotonien
 - verwendet Monotonietabellen für Kombination von Ungleichungen
 - erzeugt zusätzliche (kombinierte) Hypothesen

● Erzeugung der **Konjunktiven Normalform**

- Zu jeder Formel F in $\mathcal{A}$ gibt es in $\mathcal{A}$ eine äquivalente Formel G in KNF
- Man kann jedes Konjunkt isoliert beweisen

● **Widerspruchsbeweis**

- $\Gamma \vdash C_1 \vee \dots \vee C_n$ gültig, g.d.w. $\Gamma, \neg C_1, \dots, \neg C_n \vdash \text{ff}$ gültig

● **Ersetzen von Termen durch Variablen**

- Eine Menge von Formeln $F_i[e_1, \dots, e_k / u_1, \dots, u_k]$ in $\mathcal{A}$ ist genau dann widersprüchlich, wenn $\{F_1, \dots, F_n\}$ widersprüchlich ist

● Abtrennung der Monotonie

- **Triviale Monotonien**: Monotonieaxiome von $+$ und $-$ mit Konstanten
- **monotonicity** Regel für Anwendung nichttrivialer Monotonien
 - verwendet Monotonietabellen für Kombination von Ungleichungen
 - erzeugt zusätzliche (kombinierte) Hypothesen

● Erzeugung der **Konjunktiven Normalform**

- Zu jeder Formel F in $\mathcal{A}$ gibt es in $\mathcal{A}$ eine äquivalente Formel G in KNF
- Man kann jedes Konjunkt isoliert beweisen

● **Widerspruchsbeweis**

- $\Gamma \vdash C_1 \vee \dots \vee C_n$ gültig, g.d.w. $\Gamma, \neg C_1, \dots, \neg C_n \vdash \text{ff}$ gültig

● **Ersetzen von Termen durch Variablen**

- Eine Menge von Formeln $F_i[e_1, \dots, e_k / u_1, \dots, u_k]$ in $\mathcal{A}$ ist genau dann widersprüchlich, wenn $\{F_1, \dots, F_n\}$ widersprüchlich ist

● **Repräsentation als Ordnungsgraph**

- $\Gamma = v_1 \geq u_1 + c_1, \dots, v_n \geq u_n + c_n$ ist genau dann widersprüchlich, wenn der Ordnungsgraph $\mathcal{G}$ von Γ einen positiven Zyklus besitzt.

Anfangssequenz: $\Gamma \vdash C_1 \vee \dots \vee C_m$ (C_i atomar)

1. Transformiere Anfangssequenz in die Gestalt $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
Zerlege Konjunktionen in den C_i in Einzelannahmen (analog zu **and_e**)

Anfangssequenz: $\Gamma \vdash C_1 \vee \dots \vee C_m$ (C_i atomar)

1. Transformiere Anfangssequenz in die Gestalt $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
Zerlege Konjunktionen in den C_i in Einzelannahmen (analog zu **and_e**)
2. Transformiere Ungleichungen der Form $x \neq y$ in $x \geq y+1 \vee y \geq x+1$
Zerlege Disjunktionen (analog zu **or_e**) und betrachte Beweisziele separat

Anfangssequenz: $\Gamma \vdash C_1 \vee \dots \vee C_m$ (C_i atomar)

1. Transformiere Anfangssequenz in die Gestalt $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
Zerlege Konjunktionen in den C_i in Einzelannahmen (analog zu **and_e**)
2. Transformiere Ungleichungen der Form $x \neq y$ in $x \geq y+1 \vee y \geq x+1$
Zerlege Disjunktionen (analog zu **or_e**) und betrachte Beweisziele separat
3. Entferne Hypothesen ohne atomare elementar-arithmetische Formeln
Ersetze Teilterme, die nicht der Syntax von $\mathcal{A}$ entsprechen durch Variablen

Anfangssequenz: $\Gamma \vdash C_1 \vee \dots \vee C_m$ (C_i atomar)

1. Transformiere Anfangssequenz in die Gestalt $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
Zerlege Konjunktionen in den C_i in Einzelannahmen (analog zu **and_e**)
2. Transformiere Ungleichungen der Form $x \neq y$ in $x \geq y+1 \vee y \geq x+1$
Zerlege Disjunktionen (analog zu **or_e**) und betrachte Beweisziele separat
3. Entferne Hypothesen ohne atomare elementar-arithmetische Formeln
Ersetze Teilterme, die nicht der Syntax von $\mathcal{A}$ entsprechen durch Variablen
4. Transformiere Komparanden von Ungleichungen in Standardpolynome

Anfangssequenz: $\Gamma \vdash C_1 \vee \dots \vee C_m$ (C_i atomar)

1. Transformiere Anfangssequenz in die Gestalt $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
Zerlege Konjunktionen in den C_i in Einzelannahmen (analog zu **and_e**)
2. Transformiere Ungleichungen der Form $x \neq y$ in $x \geq y+1 \vee y \geq x+1$
Zerlege Disjunktionen (analog zu **or_e**) und betrachte Beweisziele separat
3. Entferne Hypothesen ohne atomare elementar-arithmetische Formeln
Ersetze Teilterme, die nicht der Syntax von $\mathcal{A}$ entsprechen durch Variablen
4. Transformiere Komparanden von Ungleichungen in Standardpolynome
5. Transformiere Komparanden in monadische lineare Polynome der Form $c+u_i$

Anfangssequenz: $\Gamma \vdash C_1 \vee \dots \vee C_m$ (C_i atomar)

1. Transformiere Anfangssequenz in die Gestalt $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
Zerlege Konjunktionen in den C_i in Einzelannahmen (analog zu **and_e**)
2. Transformiere Ungleichungen der Form $x \neq y$ in $x \geq y+1 \vee y \geq x+1$
Zerlege Disjunktionen (analog zu **or_e**) und betrachte Beweisziele separat
3. Entferne Hypothesen ohne atomare elementar-arithmetische Formeln
Ersetze Teilterme, die nicht der Syntax von $\mathcal{A}$ entsprechen durch Variablen
4. Transformiere Komparanden von Ungleichungen in Standardpolynome
5. Transformiere Komparanden in monadische lineare Polynome der Form $c+u_i$
6. Konvertiere Hypothesen in Ungleichungen der Gestalt $t_1 \geq t_2$
(t_1 Variable oder 0; t_2 monadisches lineares Polynom)

Anfangssequenz: $\Gamma \vdash C_1 \vee \dots \vee C_m$ (C_i atomar)

1. Transformiere Anfangssequenz in die Gestalt $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
Zerlege Konjunktionen in den C_i in Einzelannahmen (analog zu **and_e**)
2. Transformiere Ungleichungen der Form $x \neq y$ in $x \geq y+1 \vee y \geq x+1$
Zerlege Disjunktionen (analog zu **or_e**) und betrachte Beweisziele separat
3. Entferne Hypothesen ohne atomare elementar-arithmetische Formeln
Ersetze Teilterme, die nicht der Syntax von $\mathcal{A}$ entsprechen durch Variablen
4. Transformiere Komparanden von Ungleichungen in Standardpolynome
5. Transformiere Komparanden in monadische lineare Polynome der Form $c+u_i$
6. Konvertiere Hypothesen in Ungleichungen der Gestalt $t_1 \geq t_2$
(t_1 Variable oder 0; t_2 monadisches lineares Polynom)
7. Erzeuge den **Ordnungsgraphen** der Formelmenge:
Knoten für jede Variablen oder Konstante; Kante $u_i \xrightarrow{c} u_j$ repräsentiert $u_i \geq u_j + c$

Anfangssequenz: $\Gamma \vdash C_1 \vee \dots \vee C_m$ (C_i atomar)

1. Transformiere Anfangssequenz in die Gestalt $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
Zerlege Konjunktionen in den C_i in Einzelannahmen (analog zu **and_e**)
2. Transformiere Ungleichungen der Form $x \neq y$ in $x \geq y+1 \vee y \geq x+1$
Zerlege Disjunktionen (analog zu **or_e**) und betrachte Beweisziele separat
3. Entferne Hypothesen ohne atomare elementar-arithmetische Formeln
Ersetze Teilterme, die nicht der Syntax von $\mathcal{A}$ entsprechen durch Variablen
4. Transformiere Komparanden von Ungleichungen in Standardpolynome
5. Transformiere Komparanden in monadische lineare Polynome der Form $c+u_i$
6. Konvertiere Hypothesen in Ungleichungen der Gestalt $t_1 \geq t_2$
(t_1 Variable oder 0; t_2 monadisches lineares Polynom)
7. Erzeuge den **Ordnungsgraphen** der Formelmenge:
Knoten für jede Variablen oder Konstante; Kante $u_i \xrightarrow{c} u_j$ repräsentiert $u_i \geq u_j + c$
8. Teste, ob der Ordnungsgraph einen positiven Zyklus hat (Standardverfahren)
Im Erfolgsfall generiere Wohlformtheitsziele für alle “Variablen”
Bei Mißerfolg generiere Fehlermeldung

Arith: ARBEITSWEISE AM BEISPIEL

Anfangssequenz: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1 \vdash 3*x+y\geq 2*z-1$

Arith: ARBEITSWEISE AM BEISPIEL

Anfangssequenz: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1 \vdash 3*x+y\geq 2*z-1$

1. Negativform: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1$, $\neg(3*x+y\geq 2*z-1) \vdash \text{ff}$

Arith: ARBEITSWEISE AM BEISPIEL

Anfangssequenz: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1 \vdash 3*x+y\geq 2*z-1$

1. Negativform: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1$, $\neg(3*x+y\geq 2*z-1) \vdash \text{ff}$
2. Transformiere Ungleichungen der Form $x\neq y$ – *entfällt* –

Arith: ARBEITSWEISE AM BEISPIEL

Anfangssequenz: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1 \vdash 3*x+y\geq 2*z-1$

1. Negativform: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1$, $\neg(3*x+y\geq 2*z-1) \vdash \text{ff}$
2. Transformiere Ungleichungen der Form $x\neq y$ – *entfällt* –
3. Entferne Hypothesen/Ersetze nichtarithmetische Teilterme – *entfällt* –

Arith: ARBEITSWEISE AM BEISPIEL

Anfangssequenz: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1 \vdash 3*x+y\geq 2*z-1$

1. Negativform: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1$, $\neg(3*x+y\geq 2*z-1) \vdash \text{ff}$
2. Transformiere Ungleichungen der Form $x\neq y$ – *entfällt* –
3. Entferne Hypothesen/Ersetze nichtarithmetische Teilterme – *entfällt* –
4. Transformiere Komparanden in Standardpolynome

$$x+y>z, 2*x\geq z, 3*x+y\geq 1+2*z, \neg(3*x+y\geq (-1)+2*z) \vdash \text{ff}$$

Arith: ARBEITSWEISE AM BEISPIEL

Anfangssequenz: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1 \vdash 3*x+y\geq 2*z-1$

1. Negativform: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1$, $\neg(3*x+y\geq 2*z-1) \vdash \text{ff}$

2. Transformiere Ungleichungen der Form $x\neq y$ – *entfällt* –

3. Entferne Hypothesen/Ersetze nichtarithmetische Teilterme – *entfällt* –

4. Transformiere Komparanden in Standardpolynome

$$x+y>z, 2*x\geq z, 3*x+y\geq 1+2*z, \neg(3*x+y\geq (-1)+2*z) \vdash \text{ff}$$

5. Transformiere Komparanden in monadische lineare Polynome

$$u_0>z, u_1\geq z, u_2\geq 1+u_3, \neg(u_2\geq (-1)+u_3) \vdash \text{ff}$$

$$u_0 \equiv x+y, \quad u_1 \equiv 2*x, \quad u_2 \equiv 3*x+y, \quad u_3 \equiv 2*z$$

Arith: ARBEITSWEISE AM BEISPIEL

Anfangssequenz: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1 \vdash 3*x+y\geq 2*z-1$

1. Negativform: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1$, $\neg(3*x+y\geq 2*z-1) \vdash \text{ff}$

2. Transformiere Ungleichungen der Form $x\neq y$ – *entfällt* –

3. Entferne Hypothesen/Ersetze nichtarithmetische Teilterme – *entfällt* –

4. Transformiere Komparanden in Standardpolynome

$$x+y>z, 2*x\geq z, 3*x+y\geq 1+2*z, \neg(3*x+y\geq (-1)+2*z) \vdash \text{ff}$$

5. Transformiere Komparanden in monadische lineare Polynome

$$u_0>z, u_1\geq z, u_2\geq 1+u_3, \neg(u_2\geq (-1)+u_3) \vdash \text{ff}$$

$$u_0 \equiv x+y, \quad u_1 \equiv 2*x, \quad u_2 \equiv 3*x+y, \quad u_3 \equiv 2*z$$

6. Konvertiere in Ungleichungen der Gestalt $t_1\geq t_2$

$$u_0\geq z+1, u_1\geq z, u_2\geq 1+u_3, u_3\geq u_2+2 \vdash \text{ff}$$

Arith: ARBEITSWEISE AM BEISPIEL

Anfangssequenz: $x+y>z, 2*x\geq z, x+y+2*x\geq z+z+1 \vdash 3*x+y\geq 2*z-1$

1. Negativform: $x+y>z, 2*x\geq z, x+y+2*x\geq z+z+1, \neg(3*x+y\geq 2*z-1) \vdash \text{ff}$

2. Transformiere Ungleichungen der Form $x\neq y$ – *entfällt* –

3. Entferne Hypothesen/Ersetze nichtarithmetische Teilterme – *entfällt* –

4. Transformiere Komparanden in Standardpolynome

$$x+y>z, 2*x\geq z, 3*x+y\geq 1+2*z, \neg(3*x+y\geq (-1)+2*z) \vdash \text{ff}$$

5. Transformiere Komparanden in monadische lineare Polynome

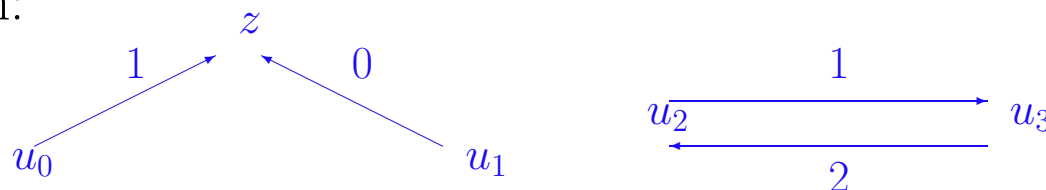
$$u_0>z, u_1\geq z, u_2\geq 1+u_3, \neg(u_2\geq (-1)+u_3) \vdash \text{ff}$$

$$u_0 \equiv x+y, \quad u_1 \equiv 2*x, \quad u_2 \equiv 3*x+y, \quad u_3 \equiv 2*z$$

6. Konvertiere in Ungleichungen der Gestalt $t_1\geq t_2$

$$u_0\geq z+1, u_1\geq z, u_2\geq 1+u_3, u_3\geq u_2+2 \vdash \text{ff}$$

7. Erzeuge den Ordnungsgraphen:



Arith: ARBEITSWEISE AM BEISPIEL

Anfangssequenz: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1 \vdash 3*x+y\geq 2*z-1$

1. Negativform: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1$, $\neg(3*x+y\geq 2*z-1) \vdash \text{ff}$

2. Transformiere Ungleichungen der Form $x\neq y$ – *entfällt* –

3. Entferne Hypothesen/Ersetze nichtarithmetische Teilterme – *entfällt* –

4. Transformiere Komparanden in Standardpolynome

$$x+y>z, 2*x\geq z, 3*x+y\geq 1+2*z, \neg(3*x+y\geq (-1)+2*z) \vdash \text{ff}$$

5. Transformiere Komparanden in monadische lineare Polynome

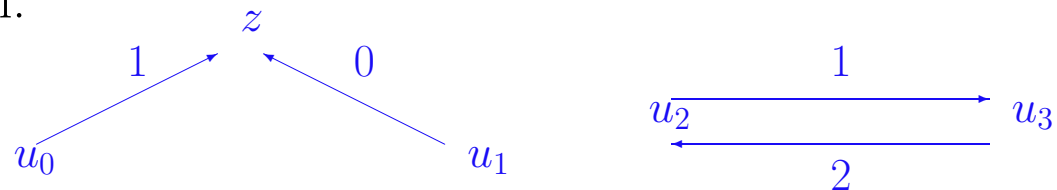
$$u_0>z, u_1\geq z, u_2\geq 1+u_3, \neg(u_2\geq (-1)+u_3) \vdash \text{ff}$$

$$u_0 \equiv x+y, \quad u_1 \equiv 2*x, \quad u_2 \equiv 3*x+y, \quad u_3 \equiv 2*z$$

6. Konvertiere in Ungleichungen der Gestalt $t_1\geq t_2$

$$u_0\geq z+1, u_1\geq z, u_2\geq 1+u_3, u_3\geq u_2+2 \vdash \text{ff}$$

7. Erzeuge den Ordnungsgraphen:



8. *Ordnungsgraph hat positiven Zyklus ...*

Arith: ARBEITSWEISE AM BEISPIEL

Anfangssequenz: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1 \vdash 3*x+y\geq 2*z-1$

1. Negativform: $x+y>z$, $2*x\geq z$, $x+y+2*x\geq z+z+1$, $\neg(3*x+y\geq 2*z-1) \vdash \text{ff}$

2. Transformiere Ungleichungen der Form $x\neq y$ – *entfällt* –

3. Entferne Hypothesen/Ersetze nichtarithmetische Teilterme – *entfällt* –

4. Transformiere Komparanden in Standardpolynome

$$x+y>z, 2*x\geq z, 3*x+y\geq 1+2*z, \neg(3*x+y\geq (-1)+2*z) \vdash \text{ff}$$

5. Transformiere Komparanden in monadische lineare Polynome

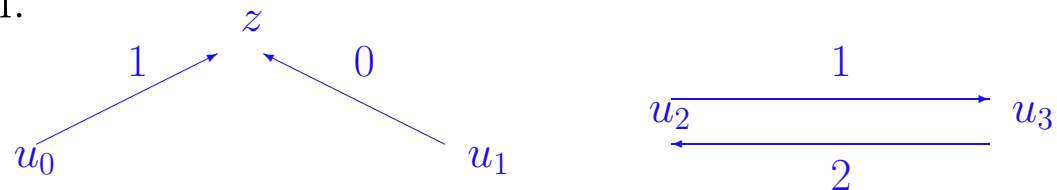
$$u_0>z, u_1\geq z, u_2\geq 1+u_3, \neg(u_2\geq (-1)+u_3) \vdash \text{ff}$$

$$u_0 \equiv x+y, \quad u_1 \equiv 2*x, \quad u_2 \equiv 3*x+y, \quad u_3 \equiv 2*z$$

6. Konvertiere in Ungleichungen der Gestalt $t_1\geq t_2$

$$u_0\geq z+1, u_1\geq z, u_2\geq 1+u_3, u_3\geq u_2+2 \vdash \text{ff}$$

7. Erzeuge den Ordnungsgraphen:



8. Ordnungsgraph hat positiven Zyklus ... **Formel ist gültig**

- Arithmetische **Komposition von Ungleichungen**

- **Arithmetische Komposition von Ungleichungen**

- z.B. **monotonicity** $i + j$ addiert Hypothesen i und j
aus $i : x+y > z$ und $j : 2*x \geq z$ entsteht $n+1 : x+y+2*x \geq z+z+1$

monotonicity: ANWENDUNG NICHTTRIVIALER MONOTONIEN

• Arithmetische Komposition von Ungleichungen

- z.B. **monotonicity** $i + j$ addiert Hypothesen i und j
 aus $i : x+y > z$ und $j : 2*x \geq z$ entsteht $n+1 : x+y+2*x \geq z+z+1$
- Neue Hypothese entsteht durch Anwendung von Monotonietabellen

Addition				
	$z > w$	$z \geq w$	$z = w$	$z \neq w$
$x > y$	$x+z \geq y+w+2$	$x+z \geq y+w+1$	$x+z \geq y+w+1$ $x+w \geq y+z+1$	-----
$x \geq y$	$x+z \geq y+w+1$	$x+z \geq y+w$	$x+z \geq y+w$ $x+w \geq y+z$	-----
$x = y$	$x+z \geq y+w+1$ $y+z \geq x+w+1$	$x+z \geq y+w$ $y+z \geq x+w$	$x+z = y+w$ $x+w = y+z$	$x+z \neq y+w$ $x+w \neq y+z$
$x \neq y$	-----	-----	$x+z \neq y+w$ $x+w \neq y+z$	-----

Multiplikation				
	$y \geq z$	$y > z$	$y = z$	$y \neq z$
$x > 0$	$x*y \geq x*z$	$x*y > x*z$	$x*y = x*z$	$x*y \neq x*z$
$x \geq 0$	$x*y \geq x*z$	$x*y \geq x*z$	$x*y = x*z$	-----
$x = 0$	$x*y = x*z$ $x*y = 0$	$x*y = x*z$ $x*y = 0$	$x*y = x*z$ $x*y = 0$	$x*y = x*z$ $x*y = 0$
$x \leq 0$	$x*y \leq x*z$	$x*y \leq x*z$	$x*y = x*z$	-----
$x < 0$	$x*y \leq x*z$	$x*y < x*z$	$x*y = x*z$	$x*y \neq x*z$
$x \neq 0$	-----	$x*y \neq x*z$	$x*y = x*z$	$x*y \neq x*z$

Subtraktion				
	$z > w$	$z \geq w$	$z = w$	$z \neq w$
$x > y$	$x-w \geq y-z+2$	$x-w \geq y-z+1$	$x-w \geq y-z+1$ $x-z \geq y-w+1$	-----
$x \geq y$	$x-w \geq y-z+1$	$x-w \geq y-z$	$x-w \geq y-z$ $x-z \geq y-w$	-----
$x = y$	$x-w \geq y-z+1$ $y-w \geq x-z+1$	$x-w \geq y-z$ $y-w \geq x-z$	$x-w = y-z$ $y-w = x-z$	$x-w \neq y-z$ $x-z \neq y-w$
$x \neq y$	-----	-----	$x-w \neq y-z$ $x-z \neq y-w$	-----

Faktorisierung				
	$x*y > x*z$	$x*y \geq x*z$	$x*y = x*z$	$x*y \neq x*z$
$x > 0$	$y > z$	$y \geq z$	$y = z$	$y \neq z$
$x < 0$	$y < z$	$y \leq z$	$y = z$	$y \neq z$
$x \neq 0$	$y \neq z$	-----	$y = z$	$y \neq z$

Entscheide lineare Ungleichungen über $\mathbb{Z}$

Entscheide lineare Ungleichungen über $\mathbb{Z}$

- Arith zu schwach für lineare Ungleichungssysteme

Entscheide lineare Ungleichungen über $\mathbb{Z}$

- Arith zu schwach für lineare Ungleichungssysteme
- Anpassung von Bledsoe's **Sup-Inf** Methode (1975)
 - Methode ist nur für rationale Zahlen korrekt und vollständig
 - Korrekt und unvollständig für $\mathbb{Z}$, aber hilfreich in der Praxis

Entscheide lineare Ungleichungen über $\mathbb{Z}$

- **Arith zu schwach** für lineare Ungleichungssysteme
- **Anpassung von Bledsoe's Sup-Inf Methode** (1975)
 - Methode ist nur für rationale Zahlen korrekt und vollständig
 - Korrekt und unvollständig für $\mathbb{Z}$, aber hilfreich in der Praxis
- **Logische Theorie: Arithmetische Formeln**
 - Kombination von Ungleichungen über arithmetischen Typen

Entscheide lineare Ungleichungen über $\mathbb{Z}$

- **Arith zu schwach** für lineare Ungleichungssysteme
- **Anpassung von Bledsoe's Sup-Inf Methode** (1975)
 - Methode ist nur für rationale Zahlen korrekt und vollständig
 - Korrekt und unvollständig für $\mathbb{Z}$, aber hilfreich in der Praxis
- **Logische Theorie: Arithmetische Formeln**
 - Kombination von Ungleichungen über arithmetischen Typen
- **Beweismethode:**
 - Extrahiere Menge linearer Ungleichungen $0 \leq e_i$
deren Unerfüllbarkeit die Gültigkeit der Sequenz impliziert

Entscheide lineare Ungleichungen über $\mathbb{Z}$

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

Entscheide lineare Ungleichungen über $\mathbb{Z}$

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

Entscheide lineare Ungleichungen über $\mathbb{Z}$

- **Arith zu schwach** für lineare Ungleichungssysteme
- **Anpassung von Bledsoe's Sup-Inf Methode** (1975)
 - Methode ist nur für rationale Zahlen korrekt und vollständig
 - Korrekt und unvollständig für $\mathbb{Z}$, aber hilfreich in der Praxis
- **Logische Theorie: Arithmetische Formeln**
 - Kombination von Ungleichungen über arithmetischen Typen
- **Beweismethode:**
 - Extrahiere Menge linearer Ungleichungen $0 \leq e_i$, deren Unerfüllbarkeit die Gültigkeit der Sequenz impliziert
 - Bestimme obere/untere Grenzen für Variablen der e_i
 - Wenn alle Variablen in $\mathbb{Z}$ erfüllbar sind, liefere Gegenbeispiel
 - Implementierung in Nuprl als ML Strategie
- **Eingebettet in die Taktik Auto'**

Analysiere Konjunktion linearer Ungleichungen über $\mathbb{Q}$

- Betrachte Formeln der Form $0 \leq e_1 \wedge \dots \wedge 0 \leq e_n$
 - e_i lineare Ausdrücke über rationalen Variablen $x_1, \dots, x_m$
 - Suche Belegung der x_j , welche die Konjunktion erfüllen

Analysiere Konjunktion linearer Ungleichungen über $\mathbb{Q}$

- **Betrachte Formeln der Form $0 \leq e_1 \wedge \dots \wedge 0 \leq e_n$**
 - e_i lineare Ausdrücke über rationalen Variablen $x_1, \dots, x_m$
 - 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 optimal (also Supremum und Infimum)
 - Erfüllende Belegung existiert, g.d.w. Infima jeweils kleiner als Suprema

Analysiere Konjunktion linearer Ungleichungen über $\mathbb{Q}$

- **Betrachte Formeln der Form $0 \leq e_1 \wedge \dots \wedge 0 \leq e_n$**
 - e_i lineare Ausdrücke über rationalen Variablen $x_1, \dots, x_m$
 - 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 optimal (also Supremum und Infimum)
 - Erfüllende Belegung existiert, g.d.w. Infima jeweils kleiner als Suprema
- **(Widerlegungs-)version für $\mathbb{Z}$ unvollständig**
 - Erfüllende Belegung über $\mathbb{Q}$ liefert nicht immer eine über $\mathbb{Z}$
 - Reparatur möglich, aber Integer Linear Programming ist $\mathcal{NP}$ -vollständig

Analysiere Konjunktion linearer Ungleichungen über $\mathbb{Q}$

- **Betrachte Formeln der Form $0 \leq e_1 \wedge \dots \wedge 0 \leq e_n$**
 - e_i lineare Ausdrücke über rationalen Variablen $x_1, \dots, x_m$
 - 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 optimal (also Supremum und Infimum)
 - Erfüllende Belegung existiert, g.d.w. Infima jeweils kleiner als Suprema
- **(Widerlegungs-)version für $\mathbb{Z}$ unvollständig**
 - Erfüllende Belegung über $\mathbb{Q}$ liefert nicht immer eine über $\mathbb{Z}$
 - Reparatur möglich, aber Integer Linear Programming ist $\mathcal{NP}$ -vollständig
 - Korrektheit: Unerfüllbarkeit über $\mathbb{Q}$ bedeutet Unerfüllbarkeit über $\mathbb{Z}$

● Arithmetische Typen

- $\mathbb{Z}$ (int), $\mathbb{N}$ (nat), $\mathbb{N}^+$ (nat_plus), $\mathbb{Z}^{-0}$ (int_nzero)
- $\{i \dots\}$ (int_upper), $\{i \dots j^-\}$ (int_seg), $\{i \dots j\}$ (int_iseg)

● Arithmetische Typen

- $\mathbb{Z}$ (int), $\mathbb{N}$ (nat), $\mathbb{N}^+$ (nat_plus), $\mathbb{Z}^{-0}$ (int_nzero)
- $\{i \dots\}$ (int_upper), $\{i \dots j^-\}$ (int_seg), $\{i \dots j\}$ (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 Typen

- $\mathbb{Z}$ (int), $\mathbb{N}$ (nat), $\mathbb{N}^+$ (nat_plus), $\mathbb{Z}^{-0}$ (int_nzero)
- $\{i \dots\}$ (int_upper), $\{i \dots j^-\}$ (int_seg), $\{i \dots j\}$ (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

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

1. Extrahiere arithmetische Formel $F = r_1 \wedge \dots \wedge r_n \wedge \neg r_0$
 - Aus Unerfüllbarkeit von F folgt Gültigkeit der Anfangssequenz

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

1. Extrahiere arithmetische Formel $F = r_1 \wedge \dots \wedge r_n \wedge \neg r_0$
 - 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

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

1. Extrahiere arithmetische Formel $F = r_1 \wedge \dots \wedge r_n \wedge \neg r_0$
 - 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 Standard-Polynom)

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

1. Extrahiere arithmetische Formel $F = r_1 \wedge \dots \wedge r_n \wedge \neg r_0$
 - 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 Standard-Polynom)
4. Ersetze nichtlineare Teilausdrücke durch Variablen

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

1. Extrahiere arithmetische Formel $F = r_1 \wedge \dots \wedge r_n \wedge \neg r_0$
 - 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 Standard-Polynom)
4. Ersetze nichtlineare Teilausdrücke durch Variablen
5. Wende Sup-Inf Basismethode auf jedes Disjunkt an
 - Wenn jedes Disjunkt unerfüllbar ist, erzeuge Wohlgeformtheitsziele
 - Andernfalls ist erfüllende Belegung ein Gegenbeispiel in “supinf_info”

Ergänze arithmetische Kontextinformation

- Extrahiere Ungleichungen aus Typinformation
 - Z.B. aus Deklaration $x:\mathbb{N}$ extrahiere $0 \leq x$

Ergänze arithmetische Kontextinformation

- **Extrahiere Ungleichungen aus Typinformation**
 - Z.B. aus Deklaration $x:\mathbb{N}$ extrahiere $0 \leq x$
 - Bestimme Typ der in den Ungleichungen vorkommenden Ausdrücke
- get_type**: (unvollständiger) Typ-Inferenz-Algorithmus in ML

Ergänze arithmetische Kontextinformation

- **Extrahiere Ungleichungen aus Typinformation**
 - Z.B. aus Deklaration $x:\mathbb{N}$ extrahiere $0 \leq x$
 - Bestimme Typ der in den Ungleichungen vorkommenden Ausdrücke
 - **get_type**: (unvollständiger) Typ-Inferenz-Algorithmus in ML
 - Ergänze Prädikat des entsprechenden Teiltyps von $\mathbb{Z}$

Ergänze arithmetische Kontextinformation

- **Extrahiere Ungleichungen aus Typinformation**

- Z.B. aus Deklaration $x:\mathbb{N}$ extrahiere $0 \leq x$
- Bestimme Typ der in den Ungleichungen vorkommenden Ausdrücke
- **get_type**: (unvollständiger) Typ-Inferenz-Algorithmus in ML
- Ergänze Prädikat des entsprechenden Teiltyps von $\mathbb{Z}$

- **Ergänze arithmetische Lemmata**

- Z.B. bei Vorkommen von $|l_1 @ l_2|$ ergänze $|l_1 @ l_2| = |l_1| + |l_2|$

Ergänze arithmetische Kontextinformation

- **Extrahiere Ungleichungen aus Typinformation**

- Z.B. aus Deklaration $x:\mathbb{N}$ extrahiere $0 \leq x$
- Bestimme Typ der in den Ungleichungen vorkommenden Ausdrücke
- **get_type**: (unvollständiger) Typ-Inferenz-Algorithmus in ML
- Ergänze Prädikat des entsprechenden Teiltyps von $\mathbb{Z}$

- **Ergänze arithmetische Lemmata**

- Z.B. bei Vorkommen von $|l_1 @ l_2|$ ergänze $|l_1 @ l_2| = |l_1| + |l_2|$
- Erlaubte Lemmata müssen global als solche deklariert sein

Ergänze arithmetische Kontextinformation

- **Extrahiere Ungleichungen aus Typinformation**

- Z.B. aus Deklaration $x:\mathbb{N}$ extrahiere $0 \leq x$
- Bestimme Typ der in den Ungleichungen vorkommenden Ausdrücke
- **get_type**: (unvollständiger) Typ-Inferenz-Algorithmus in ML
- Ergänze Prädikat des entsprechenden Teiltyps von $\mathbb{Z}$

- **Ergänze arithmetische Lemmata**

- Z.B. bei Vorkommen von $|l_1 @ l_2|$ ergänze $|l_1 @ l_2| = |l_1| + |l_2|$
- Erlaubte Lemmata müssen global als solche deklariert sein

- **Prozedur ist experimentell**

- Viele Verbesserungen möglich

GLEICHHEITSSCHLIESSEN

Folgt eine Gleichheit aus anderen Gleichheiten?

Folgt eine Gleichheit aus anderen Gleichheiten?

- Wichtig für praktische Beweisführung

- z.B.: $f(f(a, b), b) = a$ folgt aus $f(a, b) = a$

- $g(a) = a$ folgt aus $g(g(g(a))) = a$ und $g(g(g(g(g(a)))))) = a$

Folgt eine Gleichheit aus anderen Gleichheiten?

- **Wichtig für praktische Beweisführung**

- z.B.: $f(f(a, b), b) = a$ folgt aus $f(a, b) = a$
 $g(a) = a$ folgt aus $g(g(g(a))) = a$ und $g(g(g(g(g(a)))))) = a$
- Intuitiver Beweis einfach
- Regelbasierte Beweise aufwendig

Folgt eine Gleichheit aus anderen Gleichheiten?

- **Wichtig für praktische Beweisführung**

- z.B.: $f(f(a, b), b) = a$ folgt aus $f(a, b) = a$
 $g(a) = a$ folgt aus $g(g(g(a))) = a$ und $g(g(g(g(g(a)))))) = a$
- Intuitiver Beweis einfach
- Regelbasierte Beweise aufwendig

- **Elementare Gleichheit ist entscheidbar**

- Einfache Theorie: Gleichheiten mit uninterpretierten Symbolen
- Semantik: Reflexivität, Symmetrie, Transitivität, Substitution

Folgt eine Gleichheit aus anderen Gleichheiten?

- **Wichtig für praktische Beweisführung**

- z.B.: $f(f(a, b), b) = a$ folgt aus $f(a, b) = a$
 $g(a) = a$ folgt aus $g(g(g(a))) = a$ und $g(g(g(g(g(a)))))) = a$
- Intuitiver Beweis einfach
- Regelbasierte Beweise aufwendig

- **Elementare Gleichheit ist entscheidbar**

- Einfache Theorie: Gleichheiten mit uninterpretierten Symbolen
- Semantik: Reflexivität, Symmetrie, Transitivität, Substitution

- **Effiziente Verfahren verfügbar**

- Berechnung der transitiven Hülle einer Äquivalenzrelation
- Technisch: Kongruenzabschluß des Relationsgraphen

Entscheide quantorenfreie Gleichheiten

Entscheide quantorenfreie Gleichheiten

- **Anfangssequenz:** $\Gamma, E_1, \dots, E_n \vdash E_0$
 - E_i Gleichheit über einem Typ T

Entscheide quantorenfreie Gleichheiten

- **Anfangssequenz:** $\Gamma, E_1, \dots, E_n \vdash E_0$
 - E_i Gleichheit über einem Typ T
- **Logische Theorie: Gleichheitsrelationen**
 - Gleichheiten mit uninterpretierten Funktionssymbolen und Variablen
 - Reflexivität, Symmetrie, Transitivität für Elemente und Typen

Entscheide quantorenfreie Gleichheiten

- **Anfangssequenz:** $\Gamma, E_1, \dots, E_n \vdash E_0$
 - E_i Gleichheit über einem Typ T
- **Logische Theorie: Gleichheitsrelationen**
 - Gleichheiten mit uninterpretierten Funktionssymbolen und Variablen
 - Reflexivität, Symmetrie, Transitivität für Elemente und Typen
- **Beweismethode: begrenzter Kongruenzabschluß**
 - Bilde transitive Hülle der Gleichungen in den Hypothesen
 - Substitution reduziert auf taktische Dekomposition
 - Teste ob Konklusion in transitiver Hülle enthalten ist

Entscheide quantorenfreie Gleichheiten

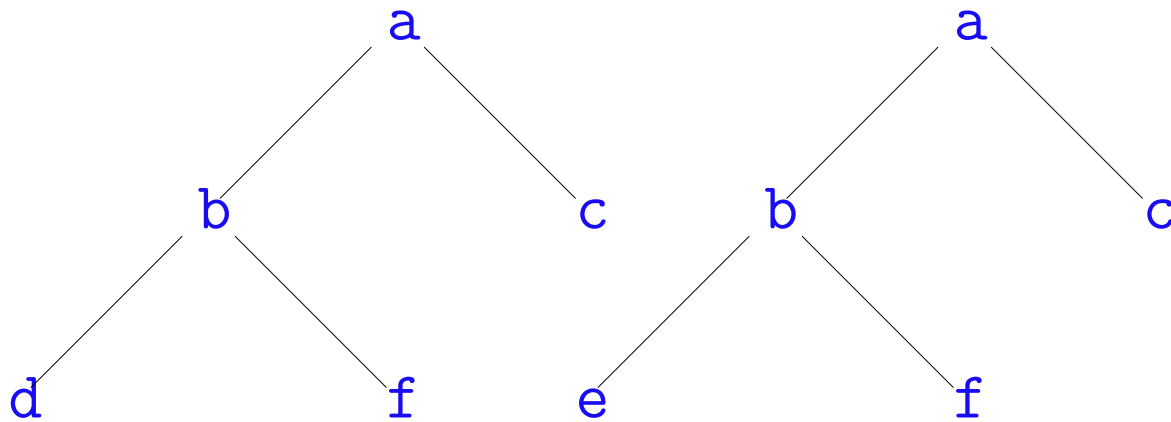
- **Anfangssequenz:** $\Gamma, E_1, \dots, E_n \vdash E_0$
 - E_i Gleichheit über einem Typ T
- **Logische Theorie: Gleichheitsrelationen**
 - Gleichheiten mit uninterpretierten Funktionssymbolen und Variablen
 - Reflexivität, Symmetrie, Transitivität für Elemente und Typen
- **Beweismethode: begrenzter Kongruenzabschluß**
 - Bilde transitive Hülle der Gleichungen in den Hypothesen
 - Substitution reduziert auf taktische Dekomposition
 - Teste ob Konklusion in transitiver Hülle enthalten ist
 - Implementierung in Nuprl als Lisp Prozedur

Entscheide quantorenfreie Gleichheiten

- **Anfangssequenz:** $\Gamma, E_1, \dots, E_n \vdash E_0$
 - E_i Gleichheit über einem Typ T
- **Logische Theorie: Gleichheitsrelationen**
 - Gleichheiten mit uninterpretierten Funktionssymbolen und Variablen
 - Reflexivität, Symmetrie, Transitivität für Elemente und Typen
- **Beweismethode: begrenzter Kongruenzabschluß**
 - Bilde transitive Hülle der Gleichungen in den Hypothesen
 - Substitution reduziert auf taktische Dekomposition
 - Teste ob Konklusion in transitiver Hülle enthalten ist
 - Implementierung in Nuprl als Lisp Prozedur
- **Eingebettet in die Taktik Auto**

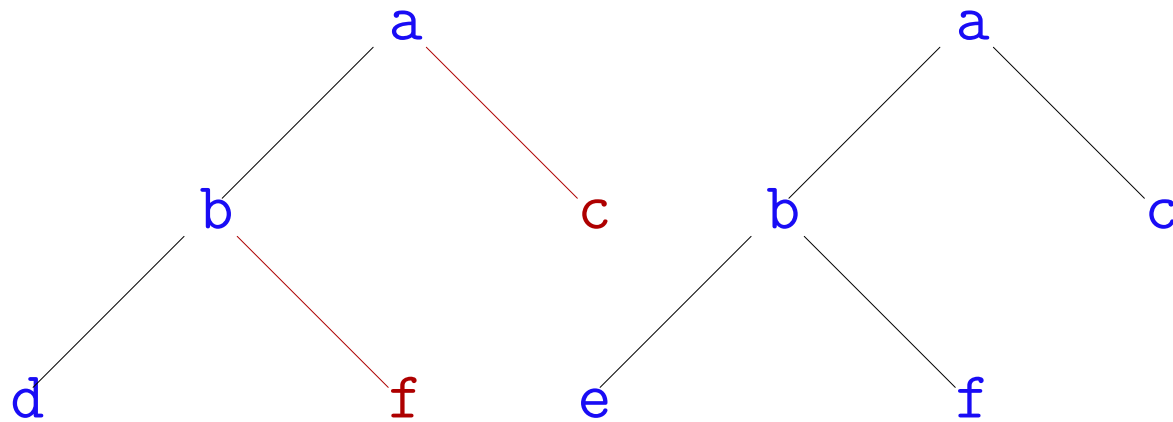
GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

Zeige : $a(b(d,f),c) = a(b(e,f),c)$ folgt aus $d=e$



GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

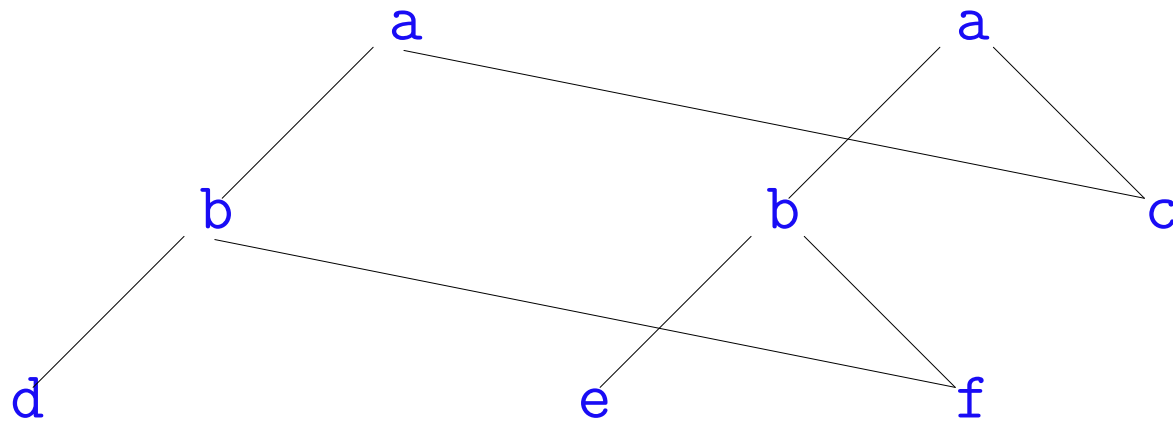
Zeige : $a(b(d,f),c) = a(b(e,f),c)$ folgt aus $d=e$



1. Verschmelze identische Knoten

GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

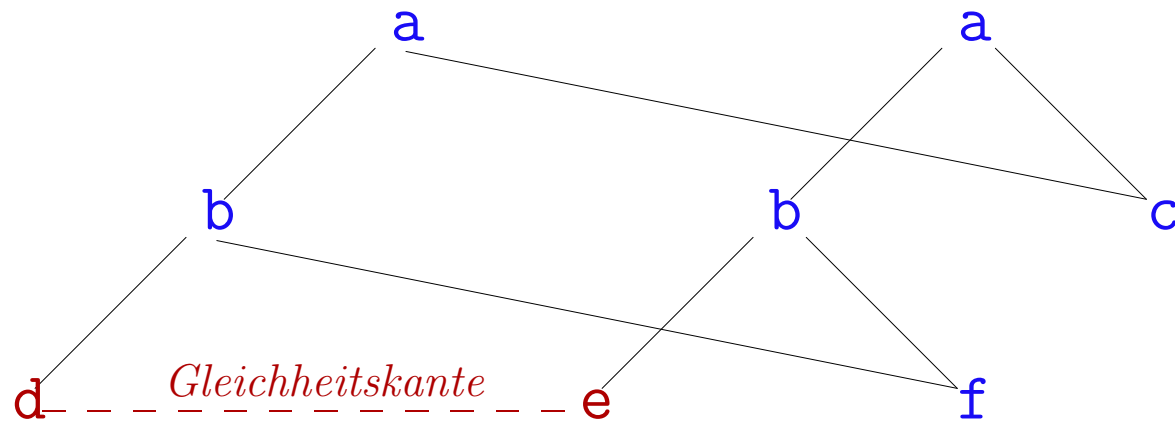
Zeige : $a(b(d,f),c) = a(b(e,f),c)$ folgt aus $d=e$



1. Verschmelze identische Knoten

GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

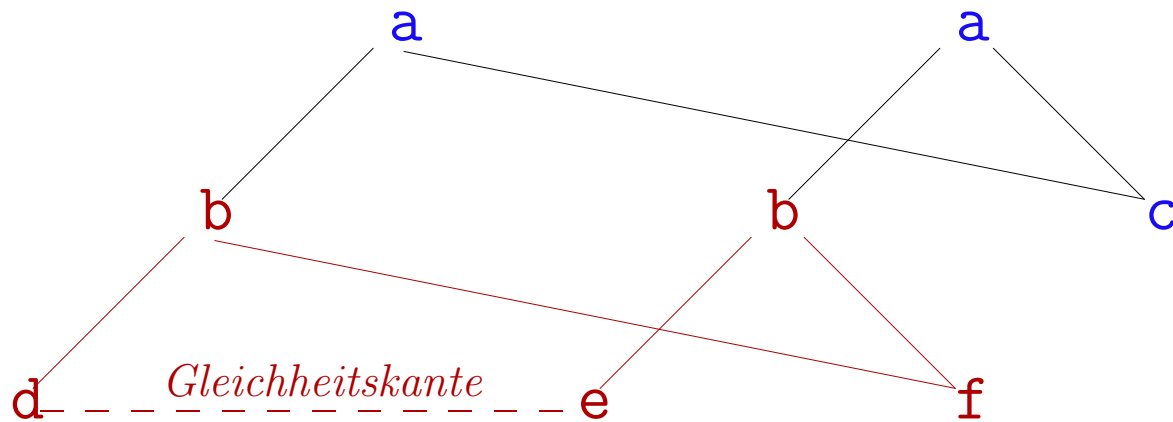
Zeige : $a(b(d,f),c) = a(b(e,f),c)$ folgt aus $d=e$



1. Verschmelze identische Knoten
2. Verbinde gleiche Knoten durch Gleichheitskante

GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

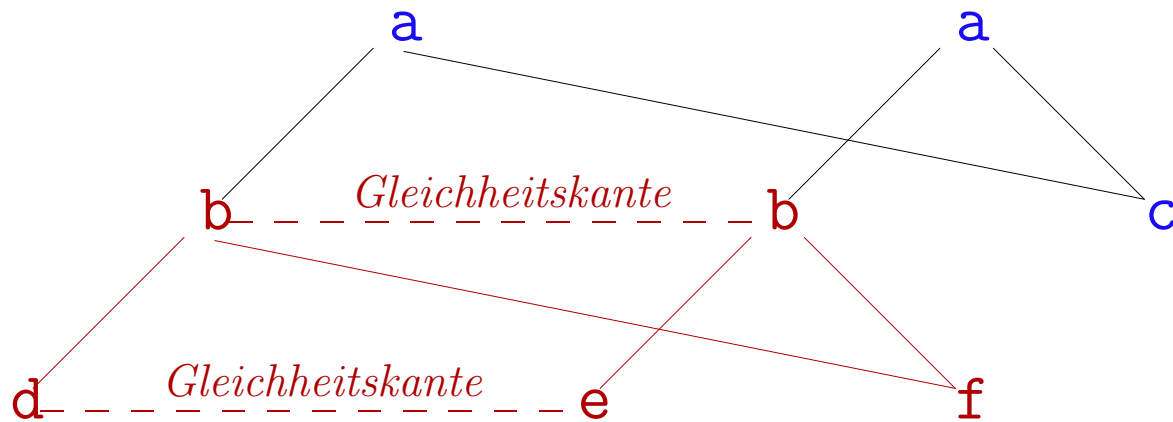
Zeige : $a(b(d,f),c) = a(b(e,f),c)$ **folgt aus** $d=e$



1. Verschmelze identische Knoten
2. Verbinde gleiche Knoten durch Gleichheitskante
3. Verbinde Wurzeln von Teilbäumen, die in allen Knoten gleich sind

GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

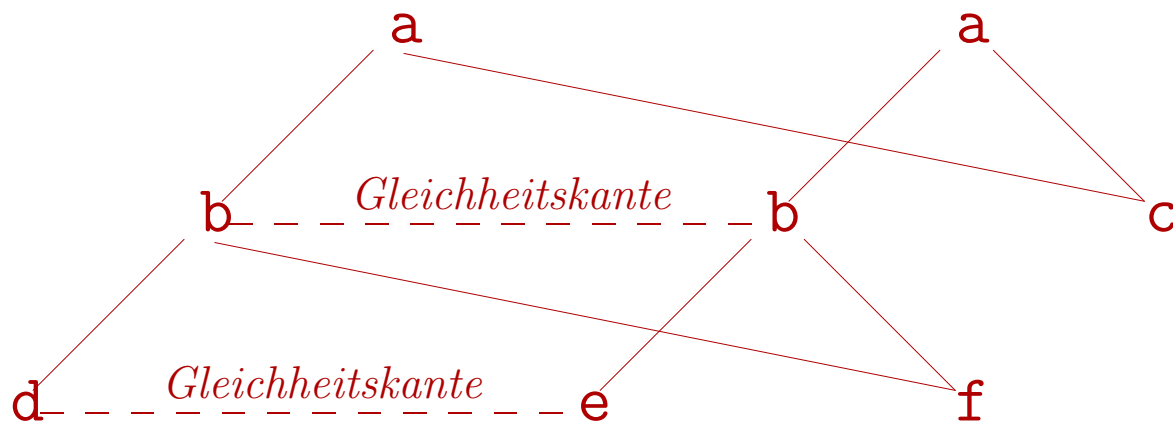
Zeige : $a(b(d,f),c) = a(b(e,f),c)$ **folgt aus** $d=e$



1. Verschmelze identische Knoten
2. Verbinde gleiche Knoten durch Gleichheitskante
3. Verbinde Wurzeln von Teilbäumen, die in allen Knoten gleich sind

GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

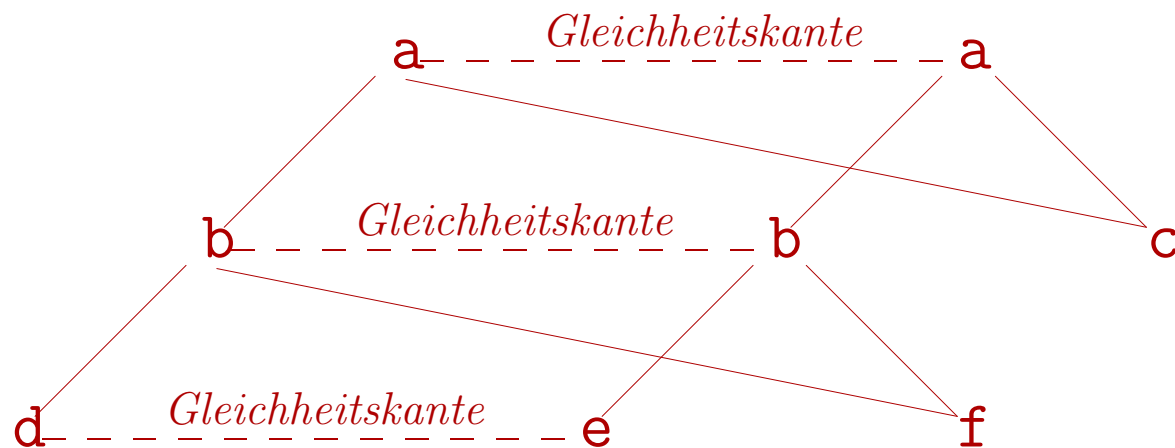
Zeige : $a(b(d,f),c) = a(b(e,f),c)$ **folgt aus** $d=e$



1. Verschmelze identische Knoten
2. Verbinde gleiche Knoten durch Gleichheitskante
3. Verbinde Wurzeln von Teilbäumen, die in allen Knoten gleich sind

GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

Zeige : $a(b(d,f),c) = a(b(e,f),c)$ **folgt aus** $d=e$



1. Verschmelze identische Knoten
2. Verbinde gleiche Knoten durch Gleichheitskante
3. Verbinde Wurzeln von Teilbäumen, die in allen Knoten gleich sind

Gleichheit $\hat{=}$ Wurzeln der Termbäume sind verbunden

- **Gerichteter Graph** $G = (V, E)$
 - $l(v)$: Markierung des Knoten v in G
 - $\delta(v)$: Anzahl der von v ausgehenden Kanten
 - $v[i]$: i -ter Nachfolgerknoten von v
 - u **Vorgänger** von v , wenn $v = u[i]$ für ein i

- **Gerichteter Graph** $G = (V, E)$
 - $l(v)$: Markierung des Knoten v in G
 - $\delta(v)$: Anzahl der von v ausgehenden Kanten
 - $v[i]$: i -ter Nachfolgerknoten von v
 - u **Vorgänger** von v , wenn $v = u[i]$ für ein i
- **Äquivalenzrelation** R auf V
 - u und v kongruent unter R ($u \sim_R v$):
$$l(u) = l(v), \quad \delta(u) = \delta(v) \quad \text{und für alle } i \quad (u[i], v[i]) \in R$$

- **Gerichteter Graph** $G = (V, E)$
 - $l(v)$: Markierung des Knoten v in G
 - $\delta(v)$: Anzahl der von v ausgehenden Kanten
 - $v[i]$: i -ter Nachfolgerknoten von v
 - u **Vorgänger** von v , wenn $v = u[i]$ für ein i
- **Äquivalenzrelation** R auf V
 - u und v kongruent unter R ($u \sim_R v$):
$$l(u) = l(v), \quad \delta(u) = \delta(v) \quad \text{und für alle } i \quad (u[i], v[i]) \in R$$
 - R abgeschlossen unter Kongruenzen: $u \sim_R v \Rightarrow (u, v) \in R$

- **Gerichteter Graph** $G = (V, E)$
 - $l(v)$: Markierung des Knoten v in G
 - $\delta(v)$: Anzahl der von v ausgehenden Kanten
 - $v[i]$: i -ter Nachfolgerknoten von v
 - u **Vorgänger** von v , wenn $v = u[i]$ für ein i
- **Äquivalenzrelation** R auf V
 - u und v kongruent unter R ($u \sim_R v$):
 $l(u) = l(v)$, $\delta(u) = \delta(v)$ und für alle i $(u[i], v[i]) \in R$
 - R abgeschlossen unter Kongruenzen: $u \sim_R v \Rightarrow (u, v) \in R$
 - **Kongruenzabschluß** R^* : eindeutige minimale Erweiterung von R , die abgeschlossen unter Kongruenzen und Äquivalenzrelation ist

- **Gerichteter Graph** $G = (V, E)$
 - $l(v)$: Markierung des Knoten v in G
 - $\delta(v)$: Anzahl der von v ausgehenden Kanten
 - $v[i]$: i -ter Nachfolgerknoten von v
 - u **Vorgänger** von v , wenn $v = u[i]$ für ein i
- **Äquivalenzrelation** R auf V
 - u und v kongruent unter R ($u \sim_R v$):
 $l(u) = l(v)$, $\delta(u) = \delta(v)$ und für alle i $(u[i], v[i]) \in R$
 - R abgeschlossen unter Kongruenzen: $u \sim_R v \Rightarrow (u, v) \in R$
 - **Kongruenzabschluß** R^* : eindeutige minimale Erweiterung von R ,
die abgeschlossen unter Kongruenzen und Äquivalenzrelation ist
 $\hat{=}$ Menge aller Äquivalenzen, die logisch aus R folgen

- **Gerichteter Graph** $G = (V, E)$
 - $l(v)$: Markierung des Knoten v in G
 - $\delta(v)$: Anzahl der von v ausgehenden Kanten
 - $v[i]$: i -ter Nachfolgerknoten von v
 - u **Vorgänger** von v , wenn $v = u[i]$ für ein i
- **Äquivalenzrelation** R auf V
 - u und v kongruent unter R ($u \sim_R v$):
 $l(u) = l(v)$, $\delta(u) = \delta(v)$ und für alle i $(u[i], v[i]) \in R$
 - R abgeschlossen unter Kongruenzen: $u \sim_R v \Rightarrow (u, v) \in R$
 - **Kongruenzabschluß** R^* : eindeutige minimale Erweiterung von R ,
die abgeschlossen unter Kongruenzen und Äquivalenzrelation ist
 $\hat{=}$ Menge aller Äquivalenzen, die logisch aus R folgen

GLEICHHEITSSCHLIESSEN ALS KONGRUENZABSCHLUSS

Folgt $s = t$ aus $s_1=t_1, \dots, s_n=t_n$?

- **Konstruiere Graph G** von $s, s_1, \dots, s_n, t, t_1, \dots, t_n$
 - G besteht aus Termbäumen von $s, s_1, \dots, s_n, t, t_1, \dots, t_n$
 - Identische Teilausdrücke werden durch denselben Teilbaum dargestellt

GLEICHHEITSSCHLIESSEN ALS KONGRUENZABSCHLUSS

Folgt $s = t$ aus $s_1=t_1, \dots, s_n=t_n$?

- **Konstruiere Graph G von $s, s_1, \dots, s_n, t, t_1, \dots, t_n$**
 - G besteht aus Termbäumen von $s, s_1, \dots, s_n, t, t_1, \dots, t_n$
 - Identische Teilausdrücke werden durch denselben Teilbaum dargestellt
- **Bestimme Kongruenzabschluß der $s_i=t_i$ iterativ**
 - Start: R ist Identitätsrelation auf den Knoten von G ($R^* = R$)
 - Im Schritt i bestimme Kongruenzabschluß von $R^* \cup \{(\tau(s_i), \tau(t_i))\}$
($\tau(u)$: Wurzelknoten des Termbaums von u)
 - Repräsentiere R^* als Menge von Äquivalenzklassen $\{[u]_R \mid u \in V\}$
($[u]_R \equiv \{x \in V \mid (x, u) \in R\}$)

GLEICHHEITSSCHLIESSEN ALS KONGRUENZABSCHLUSS

Folgt $s = t$ aus $s_1=t_1, \dots, s_n=t_n$?

- **Konstruiere Graph G von $s, s_1, \dots, s_n, t, t_1, \dots, t_n$**
 - G besteht aus Termbäumen von $s, s_1, \dots, s_n, t, t_1, \dots, t_n$
 - Identische Teilausdrücke werden durch denselben Teilbaum dargestellt
- **Bestimme Kongruenzabschluß der $s_i=t_i$ iterativ**
 - Start: R ist Identitätsrelation auf den Knoten von G ($R^* = R$)
 - Im Schritt i bestimme Kongruenzabschluß von $R^* \cup \{(\tau(s_i), \tau(t_i))\}$
($\tau(u)$: Wurzelknoten des Termbaums von u)
 - Repräsentiere R^* als Menge von Äquivalenzklassen $\{ [u]_R \mid u \in V \}$
($[u]_R \equiv \{x \in V \mid (x, u) \in R\}$)
- **Teste Äquivalenz von s und t**
 - $s = t$ gilt genau dann, wenn $(\tau(s), \tau(t)) \in R^*$

GLEICHHEITSSCHLIESSEN ALS KONGRUENZABSCHLUSS

Folgt $s = t$ aus $s_1=t_1, \dots, s_n=t_n$?

- **Konstruiere Graph G von $s, s_1, \dots, s_n, t, t_1, \dots, t_n$**
 - G besteht aus Termbäumen von $s, s_1, \dots, s_n, t, t_1, \dots, t_n$
 - Identische Teilausdrücke werden durch denselben Teilbaum dargestellt
- **Bestimme Kongruenzabschluß der $s_i=t_i$ iterativ**
 - Start: R ist Identitätsrelation auf den Knoten von G ($R^* = R$)
 - Im Schritt i bestimme Kongruenzabschluß von $R^* \cup \{(\tau(s_i), \tau(t_i))\}$
($\tau(u)$: Wurzelknoten des Termbaums von u)
 - Repräsentiere R^* als Menge von Äquivalenzklassen $\{[u]_R \mid u \in V\}$
($[u]_R \equiv \{x \in V \mid (x, u) \in R\}$)
- **Teste Äquivalenz von s und t**
 - $s = t$ gilt genau dann, wenn $(\tau(s), \tau(t)) \in R^*$

In Nuprl wegen Typbedingungen nur beschränkt einsetzbar

BERECHNE KONGRUENZABSCHLUSS VON $R \cup \{(u, v)\}$

- **Algorithmus** **MERGE**(R, u, v)

- Eingabe: gerichteter Graph $G = (V, E)$, $u, v \in V$

- Äquivalenzrelation R (abgeschlossen unter Kongruenzen)

BERECHNE KONGRUENZABSCHLUSS VON $R \cup \{(u, v)\}$

- Algorithmus **MERGE**(R, u, v)

- Eingabe: gerichteter Graph $G = (V, E)$, $u, v \in V$

- Äquivalenzrelation R (abgeschlossen unter Kongruenzen)

- Falls $u \sim_R v$, dann halte mit Ergebnis R

- Es gilt $(R \cup \{(u, v)\})^* = R$

BERECHNE KONGRUENZABSCHLUSS VON $R \cup \{(u, v)\}$

- Algorithmus **MERGE**(R, u, v)

- Eingabe: gerichteter Graph $G = (V, E)$, $u, v \in V$

- Äquivalenzrelation R (abgeschlossen unter Kongruenzen)

- Falls $u \sim_R v$, dann halte mit Ergebnis R

- Es gilt $(R \cup \{(u, v)\})^* = R$

- Andernfalls modifiziere R durch Verschmelzung

- Setze $P_u := \{x \in V \mid \exists w \in [u]_R. x \text{ Vorgänger von } w\}$

- Setze $P_v := \{x \in V \mid \exists w \in [v]_R. x \text{ Vorgänger von } w\}$

BERECHNE KONGRUENZABSCHLUSS VON $R \cup \{(u, v)\}$

- **Algorithmus** **MERGE**(R, u, v)

- Eingabe: gerichteter Graph $G = (V, E)$, $u, v \in V$

- Äquivalenzrelation R (abgeschlossen unter Kongruenzen)

- **Falls** $u \sim_R v$, **dann halte mit Ergebnis** R

- Es gilt $(R \cup \{(u, v)\})^* = R$

- **Andernfalls modifiziere** R **durch Verschmelzung**

- Setze $P_u := \{x \in V \mid \exists w \in [u]_R. x \text{ Vorgänger von } w\}$

- Setze $P_v := \{x \in V \mid \exists w \in [v]_R. x \text{ Vorgänger von } w\}$

- Vereinige Äquivalenzklassen $[u]_R$ und $[v]_R$ in R

BERECHNE KONGRUENZABSCHLUSS VON $R \cup \{(u, v)\}$

- **Algorithmus** **MERGE**(R, u, v)

- Eingabe: gerichteter Graph $G = (V, E)$, $u, v \in V$

- Äquivalenzrelation R (abgeschlossen unter Kongruenzen)

- **Falls** $u \sim_R v$, **dann halte mit Ergebnis** R

- Es gilt $(R \cup \{(u, v)\})^* = R$

- **Andernfalls modifiziere** R **durch Verschmelzung**

- Setze $P_u := \{x \in V \mid \exists w \in [u]_R. x \text{ Vorgänger von } w\}$

- Setze $P_v := \{x \in V \mid \exists w \in [v]_R. x \text{ Vorgänger von } w\}$

- **Vereinige** Äquivalenzklassen $[u]_R$ und $[v]_R$ in R

- Wiederhole für $x \in P_u$ und $y \in P_v$

- Falls $x \sim_R y$ und $[x]_R \neq [y]_R$ dann setze $R := \text{MERGE}(R, x, y)$

BERECHNE KONGRUENZABSCHLUSS VON $R \cup \{(u, v)\}$

- **Algorithmus** **MERGE**(R, u, v)

- Eingabe: gerichteter Graph $G = (V, E)$, $u, v \in V$

- Äquivalenzrelation R (abgeschlossen unter Kongruenzen)

- **Falls** $u \sim_R v$, **dann halte mit Ergebnis** R

- Es gilt $(R \cup \{(u, v)\})^* = R$

- **Andernfalls modifiziere** R **durch Verschmelzung**

- Setze $P_u := \{x \in V \mid \exists w \in [u]_R. x \text{ Vorgänger von } w\}$

- Setze $P_v := \{x \in V \mid \exists w \in [v]_R. x \text{ Vorgänger von } w\}$

- **Vereinige** Äquivalenzklassen $[u]_R$ und $[v]_R$ in R

- Wiederhole für $x \in P_u$ und $y \in P_v$

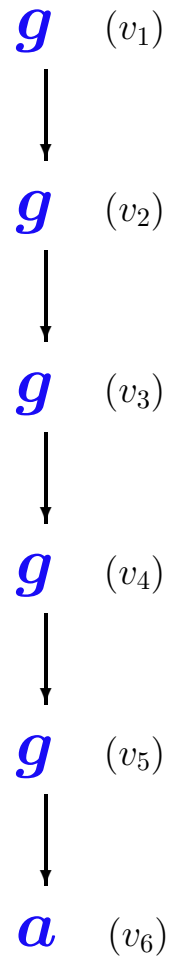
- Falls $x \sim_R y$ und $[x]_R \neq [y]_R$ dann setze $R := \text{MERGE}(R, x, y)$

- **Halte mit der modifizierten Relation** R **als Ergebnis**

KONGRUENZABSCHLUSS: $g(g(g(a))) = a$, $g(g(g(g(g(a)))))) = a$

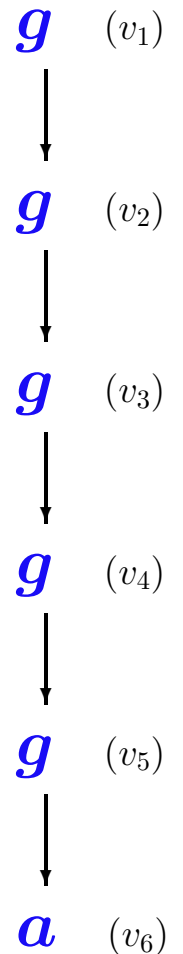
- Graph ist **Termbaum** von $g(g(g(g(g(a))))))$

- Initiale Relation: $R := \{ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\} \}$



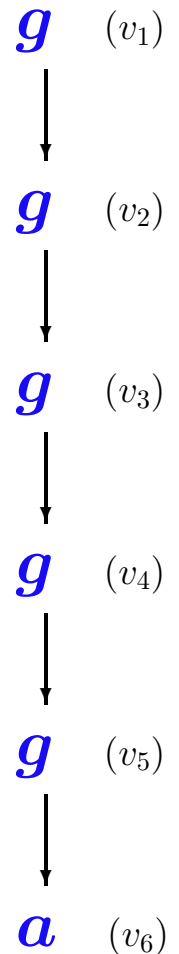
KONGRUENZABSCHLUSS: $g(g(g(a))) = a$, $g(g(g(g(g(a)))))) = a$

- Graph ist **Termbaum** von $g(g(g(g(g(a))))))$
 - Initiale Relation: $R := \{ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\} \}$
- **Hinzunahme** von $g(g(g(g(g(a)))))) = a$
 - $R := \{ \{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\} \}$ ist abgeschlossen



KONGRUENZABSCHLUSS: $g(g(g(a))) = a$, $g(g(g(g(g(a)))))) = a$

- Graph ist **Termbaum** von $g(g(g(g(g(a))))))$
 - Initiale Relation: $R := \{ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\} \}$
- Hinzunahme von $g(g(g(g(g(a)))))) = a$
 - $R := \{ \{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\} \}$ ist abgeschlossen
- Hinzunahme von $g(g(g(a))) = a$



KONGRUENZABSCHLUSS: $g(g(g(a))) = a, \quad g(g(g(g(g(a)))))) = a$

- Graph ist **Termbaum** von $g(g(g(g(g(a))))))$

- Initiale Relation: $R := \{ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\} \}$

- Hinzunahme von $g(g(g(g(g(a)))))) = a$

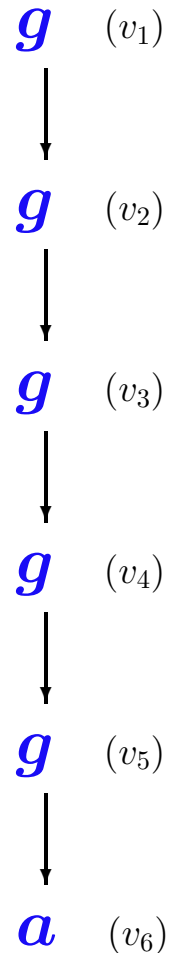
- $R := \{ \{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\} \}$ ist abgeschlossen

- Hinzunahme von $g(g(g(a))) = a$

MERGE(R, v_3, v_6):

- $P_{v_3} := \{v_2\}, \quad P_{v_6} := \{v_5\}, \quad R := \{ \{v_1, v_6, v_3\}, \{v_2\}, \{v_4\}, \{v_5\} \}$

- Wegen $(v_3, v_6) \in R$ gilt $v_2 \sim_R v_5$ aber $[v_2]_R \neq [v_5]_R$



KONGRUENZABSCHLUSS: $g(g(g(a))) = a, \quad g(g(g(g(g(a)))))) = a$

- Graph ist **Termbaum** von $g(g(g(g(g(a))))))$

- Initiale Relation: $R := \{ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\} \}$

- Hinzunahme von $g(g(g(g(g(a)))))) = a$

- $R := \{ \{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\} \}$ ist abgeschlossen

- Hinzunahme von $g(g(g(a))) = a$

MERGE(R, v_3, v_6):

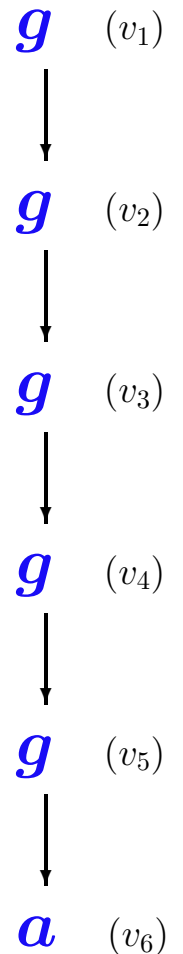
- $P_{v_3} := \{v_2\}, \quad P_{v_6} := \{v_5\}, \quad R := \{ \{v_1, v_6, v_3\}, \{v_2\}, \{v_4\}, \{v_5\} \}$

- Wegen $(v_3, v_6) \in R$ gilt $v_2 \sim_R v_5$ aber $[v_2]_R \neq [v_5]_R$

MERGE(R, v_2, v_5):

- $P_{v_2} := \{v_1\}, \quad P_{v_5} := \{v_4\}, \quad R := \{ \{v_1, v_6, v_3\}, \{v_2, v_5\}, \{v_4\} \}$

- Wegen $(v_2, v_5) \in R$ gilt $v_1 \sim_R v_4$ aber $[v_1]_R \neq [v_4]_R$



KONGRUENZABSCHLUSS: $g(g(g(a))) = a, \quad g(g(g(g(g(a)))) = a$

- Graph ist **Termbaum** von $g(g(g(g(g(a))))$

- Initiale Relation: $R := \{ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\} \}$

- **Hinzunahme von $g(g(g(g(g(a)))) = a$**

- $R := \{ \{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\} \}$ ist abgeschlossen

- **Hinzunahme von $g(g(g(a))) = a$**

MERGE(R, v_3, v_6):

- $P_{v_3} := \{v_2\}, \quad P_{v_6} := \{v_5\}, \quad R := \{ \{v_1, v_6, v_3\}, \{v_2\}, \{v_4\}, \{v_5\} \}$

- Wegen $(v_3, v_6) \in R$ gilt $v_2 \sim_R v_5$ aber $[v_2]_R \neq [v_5]_R$

MERGE(R, v_2, v_5):

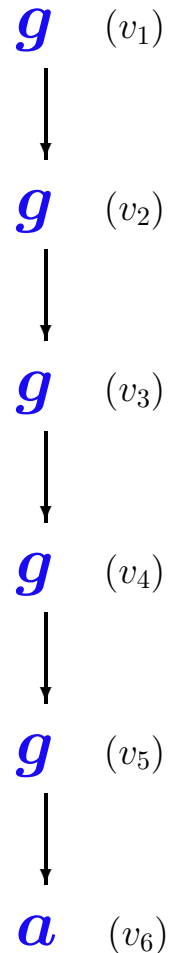
- $P_{v_2} := \{v_1\}, \quad P_{v_5} := \{v_4\}, \quad R := \{ \{v_1, v_6, v_3\}, \{v_2, v_5\}, \{v_4\} \}$

- Wegen $(v_2, v_5) \in R$ gilt $v_1 \sim_R v_4$ aber $[v_1]_R \neq [v_4]_R$

MERGE(R, v_1, v_4):

- $P_{v_1} := \{v_2, v_5\}, \quad P_{v_4} := \{v_3\}, \quad R := \{ \{v_1, v_6, v_3, v_4\}, \{v_2, v_5\} \}$

- Wegen $(v_6, v_4) \in R$ gilt $v_5 \sim_R v_3$ aber $[v_5]_R \neq [v_3]_R$



KONGRUENZABSCHLUSS: $g(g(g(a))) = a, \quad g(g(g(g(g(a)))) = a$

- Graph ist **Termbaum** von $g(g(g(g(g(a))))$

- Initiale Relation: $R := \{ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\} \}$

- **Hinzunahme von $g(g(g(g(g(a)))) = a$**

- $R := \{ \{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\} \}$ ist abgeschlossen

- **Hinzunahme von $g(g(g(a))) = a$**

MERGE(R, v_3, v_6):

- $P_{v_3} := \{v_2\}, \quad P_{v_6} := \{v_5\}, \quad R := \{ \{v_1, v_6, v_3\}, \{v_2\}, \{v_4\}, \{v_5\} \}$

- Wegen $(v_3, v_6) \in R$ gilt $v_2 \sim_R v_5$ aber $[v_2]_R \neq [v_5]_R$

MERGE(R, v_2, v_5):

- $P_{v_2} := \{v_1\}, \quad P_{v_5} := \{v_4\}, \quad R := \{ \{v_1, v_6, v_3\}, \{v_2, v_5\}, \{v_4\} \}$

- Wegen $(v_2, v_5) \in R$ gilt $v_1 \sim_R v_4$ aber $[v_1]_R \neq [v_4]_R$

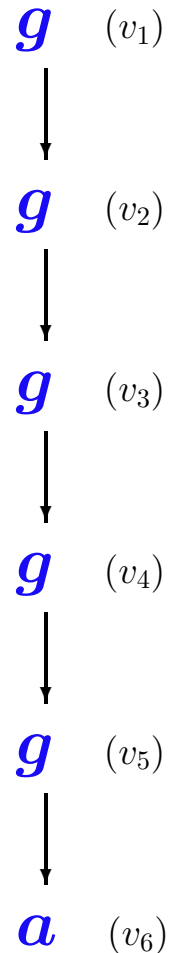
MERGE(R, v_1, v_4):

- $P_{v_1} := \{v_2, v_5\}, \quad P_{v_4} := \{v_3\}, \quad R := \{ \{v_1, v_6, v_3, v_4\}, \{v_2, v_5\} \}$

- Wegen $(v_6, v_4) \in R$ gilt $v_5 \sim_R v_3$ aber $[v_5]_R \neq [v_3]_R$

MERGE(R, v_5, v_3):

- $P_{v_5} := \{v_1, v_4\}, \quad P_{v_3} := \{v_2, v_5, v_3\}, \quad R := \{ \{v_1, v_6, v_3, v_4, v_2, v_5\} \}$



KONGRUENZABSCHLUSS: $g(g(g(a))) = a, \quad g(g(g(g(g(a)))))) = a$

- Graph ist **Termbaum** von $g(g(g(g(g(a))))))$

- Initiale Relation: $R := \{ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\} \}$

- **Hinzunahme von $g(g(g(g(g(a)))))) = a$**

- $R := \{ \{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\} \}$ ist abgeschlossen

- **Hinzunahme von $g(g(g(a))) = a$**

MERGE(R, v_3, v_6):

- $P_{v_3} := \{v_2\}, \quad P_{v_6} := \{v_5\}, \quad R := \{ \{v_1, v_6, v_3\}, \{v_2\}, \{v_4\}, \{v_5\} \}$

- Wegen $(v_3, v_6) \in R$ gilt $v_2 \sim_R v_5$ aber $[v_2]_R \neq [v_5]_R$

MERGE(R, v_2, v_5):

- $P_{v_2} := \{v_1\}, \quad P_{v_5} := \{v_4\}, \quad R := \{ \{v_1, v_6, v_3\}, \{v_2, v_5\}, \{v_4\} \}$

- Wegen $(v_2, v_5) \in R$ gilt $v_1 \sim_R v_4$ aber $[v_1]_R \neq [v_4]_R$

MERGE(R, v_1, v_4):

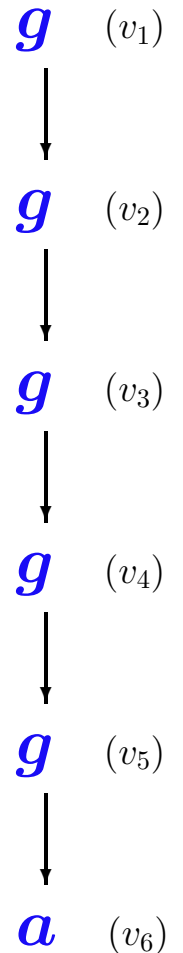
- $P_{v_1} := \{v_2, v_5\}, \quad P_{v_4} := \{v_3\}, \quad R := \{ \{v_1, v_6, v_3, v_4\}, \{v_2, v_5\} \}$

- Wegen $(v_6, v_4) \in R$ gilt $v_5 \sim_R v_3$ aber $[v_5]_R \neq [v_3]_R$

MERGE(R, v_5, v_3):

- $P_{v_5} := \{v_1, v_4\}, \quad P_{v_3} := \{v_2, v_5, v_3\}, \quad R := \{ \{v_1, v_6, v_3, v_4, v_2, v_5\} \}$

Alle Knoten sind äquivalent: $R=R^*$



- **Weitere Theorien sind effektiv entscheidbar**

- Schließen über **Listenstrukturen**
- **Geometrische Probleme**
- **Aussagenlogik** mit uninterpretierten Funktionssymbolen

- **Weitere Theorien sind effektiv entscheidbar**
 - Schließen über **Listenstrukturen**
 - **Geometrische Probleme**
 - **Aussagenlogik** mit uninterpretierten Funktionssymbolen
- **Einbettung in Typentheorie aufwendig**
 - Teilterme im Entscheidungsvorgang müssen **Typbedingungen** erfüllen
 - **Korrektheitsbeweis schwierig** zu führen

- **Weitere Theorien sind effektiv entscheidbar**
 - Schließen über **Listenstrukturen**
 - **Geometrische Probleme**
 - **Aussagenlogik** mit uninterpretierten Funktionssymbolen
- **Einbettung in Typentheorie aufwendig**
 - Teilterme im Entscheidungsvorgang müssen **Typbedingungen** erfüllen
 - **Korrektheitsbeweis schwierig** zu führen
- **Kein Ersatz für Taktik-Konzept**
 - Implementierung immer auf Systemebene
 - Benutzer kann Prozedur nicht selbst bei Bedarf erweitern
 - **Anpassungen** an Benutzerwünsche **machen Prozeduren oft unvorhersagbar**