

Automatisierte Logik und Programmierung

Einheit 13

Entscheidungsprozeduren



1

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**

- d.h. Problem ist Variation bekannter mathematischer Erkenntnisse
 - + Beweisdetails / Extraktterm ohne Bedeutung (nur Wahrheit gefragt)
 - + Formaler Beweis mit Taktiken zu aufwendig

– z.B. $x+y = y+x$

mühsame Doppelinduktion

$$y < 25 \wedge x > 22 \wedge x < y \Rightarrow y = 24$$

viele Lemmata nötig

- **Sinnvoll für “uninteressante” Beweisziele**

- d.h. Problem ist Variation bekannter mathematischer Erkenntnisse
 - + Beweisdetails / Extraktterm ohne Bedeutung (nur Wahrheit gefragt)
 - + Formaler Beweis mit Taktiken zu aufwendig

- **Möglich für algorithmisch entscheidbare Ziele**

- d.h. es gibt eine maschinennahe Charakterisierung für Gültigkeit
 - + Effizientes Entscheidungsverfahren programmierbar
 - + Schneller Test, ob Verfahren überhaupt anwendbar ist

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

- **Sinnvoll für “uninteressante” Beweisziele**

- d.h. Problem ist Variation bekannter mathematischer Erkenntnisse
 - + Beweisdetails / Extrakterm ohne Bedeutung (nur Wahrheit gefragt)
 - + Formaler Beweis mit Taktiken zu aufwendig

- **Möglich für algorithmisch entscheidbare Ziele**

- d.h. es gibt eine maschinennahe Charakterisierung für Gültigkeit
 - + Effizientes Entscheidungsverfahren programmierbar
 - + Schneller Test, ob Verfahren überhaupt anwendbar ist

- **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

ARITHMETISCHE ENTSCHEIDUNGSPROZEDUREN

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

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

$x+1 < y, 0 < t \vdash (x+1) * t < y * t$ 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 tauchen **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”*

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

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

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

- **Entscheidung nur für eingeschränkte Arithmetik**

- **Arith**: Induktionsfreie Arithmetik
- **SupInf**: Ganzzahlige lineare Ungleichungssysteme

DIE ENTSCHEIDUNGSPROZEDUR Arith

Entscheide Probleme der induktionsfreien Arithmetik

DIE ENTSCHEIDUNGSPROZEDUR Arith

Entscheide Probleme der induktionsfreien Arithmetik

- **Anfangssequenz:** $\Gamma \vdash C_1 \vee \dots \vee C_m$
 - C_i : aussagenlogische Kombination arithmetischer Relationen über $\mathbb{Z}$
 - **arithmetische Relation:** Ungleichung ($<$, $=$) mit Termen über $+$, $-$, $*$
Andersartige Terme zulässig, werden aber nicht analysiert
 - “Klausel” C_i muß quantorenfrei sein

Entscheide Probleme der induktionsfreien Arithmetik

- **Anfangssequenz:** $\Gamma \vdash C_1 \vee \dots \vee C_m$
 - C_i : aussagenlogische Kombination arithmetischer Relationen über $\mathbb{Z}$
 - **arithmetische Relation:** Ungleichung ($<$, $=$) mit Termen über $+$, $-$, $*$
Andersartige Terme zulässig, werden aber nicht analysiert
 - “Klausel” C_i muß **quantorenfrei** sein
- **Beweismethode:**
 - Transformiere Sequenz in gerichteten Graph mit gewichteten Kanten
 - Teste ob positive Zyklen im Graph vorkommen

DIE ENTSCHEIDUNGSPROZEDUR `Arith`

Entscheide Probleme der induktionsfreien Arithmetik

- **Anfangssequenz:** $\Gamma \vdash C_1 \vee \dots \vee C_m$
 - C_i : aussagenlogische Kombination arithmetischer Relationen über $\mathbb{Z}$
 - **arithmetische Relation**: Ungleichung ($<$, $=$) mit Termen über $+$, $-$, $*$
Andersartige Terme zulässig, werden aber nicht analysiert
 - “Klausel” C_i muß **quantorenfrei** sein
- **Beweismethode:**
 - Transformiere Sequenz in gerichteten Graph mit gewichteten Kanten
 - Teste ob positive Zyklen im Graph vorkommen
- **Implementierung in Nuprl als Inferenzregel**
 - Regelobjekt für **arith** verweist auf Systemprozedur der Lisp-Ebene
 - Eingebettet in die Taktik **Auto**

Arith: ARBEITSWEISE AM BEISPIEL

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

1. Erzeuge Formeln für Widerspruchsbeweis:

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

Arith: ARBEITSWEISE AM BEISPIEL

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

1. Erzeuge Formeln für Widerspruchsbeweis:

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

2. Transformiere Komparanden in Standardpolynome

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

Arith: ARBEITSWEISE AM BEISPIEL

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

1. Erzeuge Formeln für Widerspruchsbeweis:

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

2. Transformiere Komparanden in Standardpolynome

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

3. Transformiere Komparanden in monadische lineare Polynome

$$x+y \mapsto u_0, \quad 2*x \mapsto u_1, \quad 3*x+y \mapsto u_2, \quad 2*z \mapsto u_3$$
$$u_0>z, \quad u_1\geq z, \quad u_2\geq 1+u_3, \quad \neg(u_2\geq (-1)+u_3) \vdash \text{ff}$$

Arith: ARBEITSWEISE AM BEISPIEL

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

1. Erzeuge Formeln für Widerspruchsbeweis:

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

2. Transformiere Komparanden in Standardpolynome

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

3. Transformiere Komparanden in monadische lineare Polynome

$$x+y \mapsto u_0, \quad 2*x \mapsto u_1, \quad 3*x+y \mapsto u_2, \quad 2*z \mapsto u_3$$
$$u_0>z, \quad u_1\geq z, \quad u_2\geq 1+u_3, \quad \neg(u_2\geq (-1)+u_3) \vdash \text{ff}$$

4. Normiere zu Ungleichungen der Gestalt $t_1\geq t_2$

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

Arith: ARBEITSWEISE AM BEISPIEL

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

1. Erzeuge Formeln für Widerspruchsbeweis:

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

2. Transformiere Komparanden in Standardpolynome

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

3. Transformiere Komparanden in monadische lineare Polynome

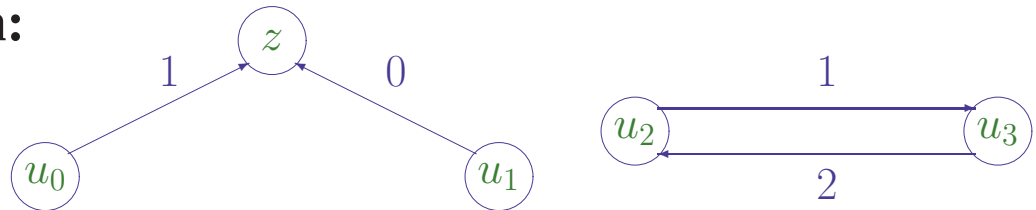
$$x+y \mapsto u_0, \quad 2*x \mapsto u_1, \quad 3*x+y \mapsto u_2, \quad 2*z \mapsto u_3$$

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

4. Normiere zu Ungleichungen der Gestalt $t_1 \geq t_2$

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

5. Erzeuge Ordnungsgraphen:



Arith: ARBEITSWEISE AM BEISPIEL

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

1. Erzeuge Formeln für Widerspruchsbeweis:

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

2. Transformiere Komparanden in Standardpolynome

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

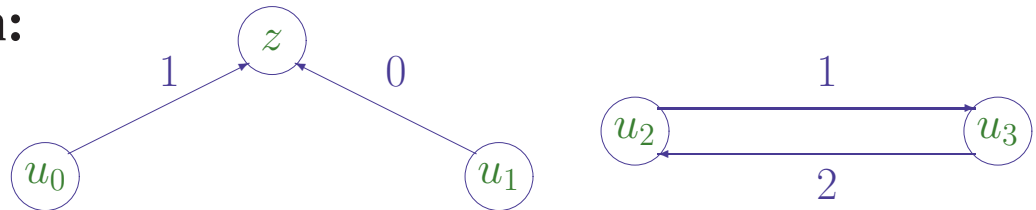
3. Transformiere Komparanden in monadische lineare Polynome

$$x+y \mapsto u_0, \quad 2*x \mapsto u_1, \quad 3*x+y \mapsto u_2, \quad 2*z \mapsto u_3$$
$$u_0 > z, \quad u_1 \geq z, \quad u_2 \geq 1+u_3, \quad \neg(u_2 \geq (-1)+u_3) \vdash \text{ff}$$

4. Normiere zu Ungleichungen der Gestalt $t_1 \geq t_2$

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

5. Erzeuge Ordnungsgraphen:



Ordnungsgraph hat positiven Zyklus ...

Arith: ARBEITSWEISE AM BEISPIEL

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

1. Erzeuge Formeln für Widerspruchsbeweis:

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

2. Transformiere Komparanden in Standardpolynome

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

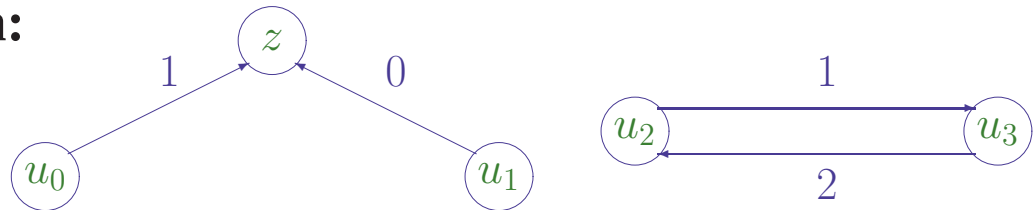
3. Transformiere Komparanden in monadische lineare Polynome

$$x+y \mapsto u_0, \quad 2*x \mapsto u_1, \quad 3*x+y \mapsto u_2, \quad 2*z \mapsto u_3$$
$$u_0 > z, \quad u_1 \geq z, \quad u_2 \geq 1+u_3, \quad \neg(u_2 \geq (-1)+u_3) \vdash \text{ff}$$

4. Normiere zu Ungleichungen der Gestalt $t_1 \geq t_2$

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

5. Erzeuge Ordnungsgraphen:



Ordnungsgraph hat positiven Zyklus ... Sequenz ist gültig

Arith: THEORETISCHES FUNDAMENT

- **Theorie *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

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

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

- **Syntax: elementar-arithmetische Formeln**

- **Syntax: elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen **Konstanten**, **Variablen** und $+$, $-$, $*$
Andersartige Terme gelten als unspezifizierte Konstanten

- **Syntax: elementar-arithmetische Formeln**

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

- **Syntax: elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen Konstanten, Variablen und $+$, $-$, $*$
Andersartige Terme gelten als unspezifizierte 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

- **Syntax: elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen Konstanten, Variablen und $+$, $-$, $*$
Andersartige Terme gelten als unspezifizierte 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)

- **Syntax: elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen Konstanten, Variablen und $+$, $-$, $*$
Andersartige Terme gelten als unspezifizierte 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

$$x=x$$

(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 gelten als unspezifizierte 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

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

(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 gelten als unspezifizierte 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

$$\neg(x < x)$$

(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 gelten als unspezifizierte 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

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

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

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

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

- **Syntax: elementar-arithmetische Formeln**

- **Terme** aufgebaut aus ganzzahligen Konstanten, Variablen und $+$, $-$, $*$
Andersartige Terme gelten als unspezifizierte 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

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

(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 gelten als unspezifizierte 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
 $1+1=2, 2+1=3, 3+1=4, \dots$

GRUNDLAGEN EINER EFFIZIENTEN ENTSCHEIDUNG

- **Entscheidbarkeit ermöglicht Widerspruchsbeweise**
 - $\Gamma \vdash C_1 \vee \dots \vee C_n$ *ist gültig in $\mathcal{A}$, g.d.w.* $\Gamma, \neg C_1, \dots, \neg C_n \vdash \text{ff}$ *gültig*

GRUNDLAGEN EINER EFFIZIENTEN ENTSCHEIDUNG

- **Entscheidbarkeit ermöglicht Widerspruchsbeweise**
 - $\Gamma \vdash C_1 \vee \dots \vee C_n$ ist gültig in $\mathcal{A}$, g.d.w. $\Gamma, \neg C_1, \dots, \neg C_n \vdash \text{ff}$ gültig
- **Entscheidbarkeit erlaubt konjunktiven Normalformen**
 - Zu jeder Formel F in $\mathcal{A}$ gibt es eine äquivalente Formel G in KNF

GRUNDLAGEN EINER EFFIZIENTEN ENTSCHEIDUNG

- **Entscheidbarkeit ermöglicht Widerspruchsbeweise**
 - $\Gamma \vdash C_1 \vee \dots \vee C_n$ ist gültig in $\mathcal{A}$, g.d.w. $\Gamma, \neg C_1, \dots, \neg C_n \vdash \text{ff}$ gültig
- **Entscheidbarkeit erlaubt konjunktiven Normalformen**
 - 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

GRUNDLAGEN EINER EFFIZIENTEN ENTSCHEIDUNG

- **Entscheidbarkeit ermöglicht Widerspruchsbeweise**
 - $\Gamma \vdash C_1 \vee \dots \vee C_n$ ist gültig in $\mathcal{A}$, g.d.w. $\Gamma, \neg C_1, \dots, \neg C_n \vdash \text{ff}$ gültig
- **Entscheidbarkeit erlaubt konjunktiven Normalformen**
 - 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**
 - $\Gamma = v_1 \geq u_1 + c_1, \dots, v_n \geq u_n + c_n$ 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 SEQUENZ $\Gamma \vdash C_1 \vee \dots \vee C_m$

1. Transformiere Sequenz in Liste atomarer arithmetischer Formeln

- Umwandlung in Widerspruchsbeweis $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
- Zerlege Konjunktionen in den C_i in atomare Formeln (wie mit andE)
- Entferne Formeln, die keine arithmetischen Ungleichungen sind
- Ersetze nichtarithmetische Ausdrücke in Termen durch Variablen

Arith: VERARBEITUNG EINER SEQUENZ $\Gamma \vdash C_1 \vee \dots \vee C_m$

1. Transformiere Sequenz in Liste atomarer arithmetischer Formeln

- Umwandlung in Widerspruchsbeweis $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
- Zerlege Konjunktionen in den C_i in atomare Formeln (wie mit **andE**)
- 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 (wie mit **orE**) und betrachte jede Konjunktion separat

Arith: VERARBEITUNG EINER SEQUENZ $\Gamma \vdash C_1 \vee \dots \vee C_m$

1. Transformiere Sequenz in Liste atomarer arithmetischer Formeln

- Umwandlung in Widerspruchsbeweis $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
- Zerlege Konjunktionen in den C_i in atomare Formeln (wie mit **andE**)
- 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 (wie mit **orE**) 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

Arith: VERARBEITUNG EINER SEQUENZ $\Gamma \vdash C_1 \vee \dots \vee C_m$

1. Transformiere Sequenz in Liste atomarer arithmetischer Formeln

- Umwandlung in Widerspruchsbeweis $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
- Zerlege Konjunktionen in den C_i in atomare Formeln (wie mit **andE**)
- 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 (wie mit **orE**) 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 Θ ; t_2 monadisches lineares Polynom

Arith: VERARBEITUNG EINER SEQUENZ $\Gamma \vdash C_1 \vee \dots \vee C_m$

1. Transformiere Sequenz in Liste atomarer arithmetischer Formeln

- Umwandlung in Widerspruchsbeweis $\Gamma, \neg C_1, \dots, \neg C_m \vdash \text{ff}$
- Zerlege Konjunktionen in den C_i in atomare Formeln (wie mit andE)
- 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 (wie mit orE) 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: STÄRKEN UND SCHWÄCHEN

- **Konsistenz mit CTT leicht nachzuweisen**
 - Aussagen, die gültig in $\mathcal{A}$ sind, sind auch gültig in CTT
 - Terme in elementar arithmetischen Formeln müssen Typ $\mathbb{Z}$ haben
 - Formulierbar als CTT-‘Regel’, die Wohlgeformtheitsziele erzeugt

Arith: STÄRKEN UND SCHWÄCHEN

- **Konsistenz mit CTT leicht nachzuweisen**
 - Aussagen, die gültig in $\mathcal{A}$ sind, sind auch gültig in CTT
 - Terme in elementar arithmetischen Formeln müssen Typ $\mathbb{Z}$ haben
 - Formulierbar als CTT-‘Regel’, die Wohlgeformtheitsziele erzeugt
- **Effizient ausführbar**
 - Es gibt Standardalgorithmen für Zyklensuche in gewichteten Graphen
 - Exponentielle worst-case Komplexität (in $\neq$) praktisch unbedeutend

Arith: STÄRKEN UND SCHWÄCHEN

- **Konsistenz mit CTT leicht nachzuweisen**

- Aussagen, die gültig in $\mathcal{A}$ sind, sind auch gültig in CTT
- Terme in elementar arithmetischen Formeln müssen Typ $\mathbb{Z}$ haben
- Formulierbar als CTT-‘Regel’, die Wohlgeformtheitsziele erzeugt

- **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
 $x \geq y \wedge z \geq w \Rightarrow x+z \geq y+w$ mit Integerkonstanten z und w
- Nichttriviale Monotonien müssen separat behandelt werden
(Monotonie von $+/-$ mit Variablen oder Monotonien mit $*$)

Arith: STÄRKEN UND SCHWÄCHEN

- **Konsistenz mit CTT leicht nachzuweisen**

- Aussagen, die gültig in $\mathcal{A}$ sind, sind auch gültig in CTT
- Terme in elementar arithmetischen Formeln müssen Typ $\mathbb{Z}$ haben
- Formulierbar als CTT-‘Regel’, die Wohlformtheitsziele erzeugt

- **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
 $x \geq y \wedge z \geq w \Rightarrow x+z \geq y+w$ mit Integerkonstanten z und w
- Nichttriviale Monotonien müssen separat behandelt werden
(Monotonie von $+/-$ mit Variablen oder Monotonien mit $*$)

- **Zu schwach für lineare Ungleichungssysteme**

- Monadische Polynome zerstören Bezüge zwischen Ungleichungen

monotonicity: BEHANDLUNG VON MONOTONIEN

- **Arithmetische Komposition von Ungleichungen**

- Erzeuge neue Hypothesen durch Anwendung von Monotonieaxiomen
- z.B. Addition von $x+y > z$ und $2*x \geq z$ ergibt $x+y+2*x \geq z+z+1$
Multiplikation von $x+y > z$, $2*x \geq 0$ ergibt $2*x^2+2*x*y \geq 2*x*z$

monotonicity: BEHANDLUNG VON MONOTONIEN

- **Arithmetische Komposition von Ungleichungen**

- Erzeuge neue Hypothesen durch Anwendung von Monotonieaxiomen
- z.B. Addition von $x+y > z$ und $2*x \geq z$ ergibt $x+y+2*x \geq z+z+1$
- Multiplikation von $x+y > z$, $2*x \geq 0$ ergibt $2*x^2+2*x*y \geq 2*x*z$

- **monotonicity-Regel operiert auf Hypothesen**

- z.B. **monotonicity** $i+j$ addiert Hypothesen i und j

monotonicity: BEHANDLUNG VON MONOTONIEN

- **Arithmetische Komposition von Ungleichungen**

- Erzeuge neue Hypothesen durch Anwendung von Monotonieaxiomen
- z.B. Addition von $x+y > z$ und $2*x \geq z$ ergibt $x+y+2*x \geq z+z+1$
- Multiplikation von $x+y > z$, $2*x \geq 0$ ergibt $2*x^2+2*x*y \geq 2*x*z$

- **monotonicity-Regel operiert auf Hypothesen**

- z.B. **monotonicity** $i+j$ addiert Hypothesen i und j

- **Effizient durch Verwendung von Monotonietabellen**

- Tabellen beschreiben Kombination verschiedenartiger Ungleichungen

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

monotonicity: MONOTONIETABELLEN

Subtraktion				
	$z > w$	$z \geq w$	$z = w$	$z \neq w$
$x > y$	$x - w \geq y - z + 2$	$x - w \geq y - z + 1$	$x - w \geq y - z + 1$ $x - z \geq y - w + 1$	-----
$x \geq y$	$x - w \geq y - z + 1$	$x - w \geq y - z$	$x - w \geq y - z$ $x - z \geq y - w$	-----
$x = y$	$x - w \geq y - z + 1$ $y - w \geq x - z + 1$	$x - w \geq y - z$ $y - w \geq x - z$	$x - w = y - z$ $y - w = x - z$	$x - w \neq y - z$ $x - z \neq y - w$
$x \neq y$	-----	-----	$x - w \neq y - z$ $x - z \neq y - w$	-----
Multiplikation				
	$y \geq z$	$y > z$	$y = z$	$y \neq z$
$x > 0$	$x * y \geq x * z$	$x * y > x * z$	$x * y = x * z$	$x * y \neq x * z$
$x \geq 0$	$x * y \geq x * z$	$x * y \geq x * z$	$x * y = x * z$	-----
$x = 0$	$x * y = x * z$ $x * y = 0$	$x * y = x * z$ $x * y = 0$	$x * y = x * z$ $x * y = 0$	$x * y = x * z$ $x * y = 0$
$x \leq 0$	$x * y \leq x * z$	$x * y \leq x * z$	$x * y = x * z$	-----
$x < 0$	$x * y \leq x * z$	$x * y < x * z$	$x * y = x * z$	$x * y \neq x * z$
$x \neq 0$	-----	$x * y \neq x * z$	$x * y = x * z$	$x * y \neq x * z$
Faktorisierung				
	$x * y > x * z$	$x * y \geq x * z$	$x * y = x * z$	$x * y \neq x * z$
$x > 0$	$y > z$	$y \geq z$	$y = z$	$y \neq z$
$x < 0$	$y < z$	$y \leq z$	$y = z$	$y \neq z$
$x \neq 0$	$y \neq z$	-----	$y = z$	$y \neq z$

DIE ENTSCHEIDUNGSPROZEDUR **SupInf**

- **Entscheide lineare Ungleichungen über $\mathbb{Z}$**
 - Sinnvoll in Anwendungen für die **Arith** zu schwach ist

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

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

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
- **Implementierung in Nuprl als ML Strategie:**
 - Bledsoe's Methode ist **nur für rationale Zahlen** korrekt und vollständig
 - **SupInf** ist korrekt, unvollständig für $\mathbb{Z}$, aber **hilfreich in der Praxis**
 - Eingebettet in die Taktik **Auto'**

DIE SUP-INF BASISMETHODE

Analysiere lineare Ungleichungsmengen über $\mathbb{Q}$

DIE SUP-INF BASISMETHODE

Analysiere lineare Ungleichungsmengen 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

DIE SUP-INF BASISMETHODE

Analysiere lineare Ungleichungsmengen 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 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

DIE SUP-INF BASISMETHODE

Analysiere lineare Ungleichungsmengen 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 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: $x+z < 5, 3*z \geq y+8, x < y \vdash 2+z > 2*y \vee x > z-5$

SupInf: ARBEITSWEISE AM BEISPIEL

1. Anfangssequenz: $x+z < 5, 3*z \geq y+8, x < y \vdash 2+z > 2*y \vee x > z-5$

2. Extrahiere arithmetische Formel für Widerspruchsbeweis

$$x+z < 5 \wedge 3*z \geq y+8 \wedge x < y \wedge \neg(2+z > 2*y) \wedge \neg(x > z-5) \vdash \text{ff}$$

SupInf: ARBEITSWEISE AM BEISPIEL

1. Anfangssequenz: $x+z < 5, 3*z \geq y+8, x < y \vdash 2+z > 2*y \vee x > z-5$

2. Extrahiere arithmetische Formel für Widerspruchsbeweis

$$x+z < 5 \wedge 3*z \geq y+8 \wedge x < y \wedge \neg(2+z > 2*y) \wedge \neg(x > z-5) \vdash \text{ff}$$

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

$$x+z+1 \leq 5 \wedge y+8 \leq 3*z \wedge x+1 \leq y \wedge 2+z \leq 2*y \wedge x \leq z-5$$

SupInf: ARBEITSWEISE AM BEISPIEL

1. **Anfangssequenz:** $x+z < 5, 3*z \geq y+8, x < y \vdash 2+z > 2*y \vee x > z-5$

2. **Extrahiere arithmetische Formel für Widerspruchsbeweis**

$$x+z < 5 \wedge 3*z \geq y+8 \wedge x < y \wedge \neg(2+z > 2*y) \wedge \neg(x > z-5) \vdash \text{ff}$$

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

$$x+z+1 \leq 5 \wedge y+8 \leq 3*z \wedge x+1 \leq y \wedge 2+z \leq 2*y \wedge x \leq z-5$$

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

$$0 \leq 4-x-z \wedge 0 \leq 3*z-y-8 \wedge 0 \leq y-x-1 \wedge 0 \leq 2*y-z-2 \wedge 0 \leq z-x-5$$

SupInf: ARBEITSWEISE AM BEISPIEL

1. **Anfangssequenz:** $x+z < 5, 3*z \geq y+8, x < y \vdash 2+z > 2*y \vee x > z-5$

2. **Extrahiere arithmetische Formel für Widerspruchsbeweis**

$$x+z < 5 \wedge 3*z \geq y+8 \wedge x < y \wedge \neg(2+z > 2*y) \wedge \neg(x > z-5) \vdash \text{ff}$$

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

$$x+z+1 \leq 5 \wedge y+8 \leq 3*z \wedge x+1 \leq y \wedge 2+z \leq 2*y \wedge x \leq z-5$$

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

$$0 \leq 4-x-z \wedge 0 \leq 3*z-y-8 \wedge 0 \leq y-x-1 \wedge 0 \leq 2*y-z-2 \wedge 0 \leq z-x-5$$

5. **Wende iterativ die Sup-Inf Basismethode an**

$$\text{SUP}(x) = -3 \quad \text{INF}(x) = -\infty, \quad \text{SUP}(y) = 1 \quad \text{INF}(y) = 14/5$$

SupInf: ARBEITSWEISE AM BEISPIEL

1. **Anfangssequenz:** $x+z < 5, 3*z \geq y+8, x < y \vdash 2+z > 2*y \vee x > z-5$

2. **Extrahiere arithmetische Formel für Widerspruchsbeweis**

$$x+z < 5 \wedge 3*z \geq y+8 \wedge x < y \wedge \neg(2+z > 2*y) \wedge \neg(x > z-5) \vdash \text{ff}$$

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

$$x+z+1 \leq 5 \wedge y+8 \leq 3*z \wedge x+1 \leq y \wedge 2+z \leq 2*y \wedge x \leq z-5$$

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

$$0 \leq 4-x-z \wedge 0 \leq 3*z-y-8 \wedge 0 \leq y-x-1 \wedge 0 \leq 2*y-z-2 \wedge 0 \leq z-x-5$$

5. **Wende iterativ die Sup-Inf Basismethode an**

$$\text{SUP}(x) = -3 \quad \text{INF}(x) = -\infty, \quad \text{SUP}(y) = 1 \quad \text{INF}(y) = 14/5$$

6. $\text{SUP}(y) < \text{INF}(y) \dots$

SupInf: ARBEITSWEISE AM BEISPIEL

1. **Anfangssequenz:** $x+z < 5, 3*z \geq y+8, x < y \vdash 2+z > 2*y \vee x > z-5$

2. **Extrahiere arithmetische Formel für Widerspruchsbeweis**

$$x+z < 5 \wedge 3*z \geq y+8 \wedge x < y \wedge \neg(2+z > 2*y) \wedge \neg(x > z-5) \vdash \text{ff}$$

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

$$x+z+1 \leq 5 \wedge y+8 \leq 3*z \wedge x+1 \leq y \wedge 2+z \leq 2*y \wedge x \leq z-5$$

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

$$0 \leq 4-x-z \wedge 0 \leq 3*z-y-8 \wedge 0 \leq y-x-1 \wedge 0 \leq 2*y-z-2 \wedge 0 \leq z-x-5$$

5. **Wende iterativ die Sup-Inf Basismethode an**

$$\text{SUP}(x) = -3 \quad \text{INF}(x) = -\infty, \quad \text{SUP}(y) = 1 \quad \text{INF}(y) = 14/5$$

6. $\text{SUP}(y) < \text{INF}(y) \dots$ **Sequenz ist gültig**

– Arithmetische Formel kann nicht erfüllt werden

SupInf: WIDERLEGUNGSBEISPIEL

1. Anfangssequenz:

$$x < 3 * y + 2, \quad x = 1 \vdash x = y$$

SupInf: WIDERLEGUNGSBEISPIEL

1. Anfangssequenz:

$$x < 3 * y + 2, \quad x = 1 \vdash x = y$$

2. Extrahiere arithmetische Formel für Widerspruchsbeweis

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

SupInf: WIDERLEGUNGSBEISPIEL

1. Anfangssequenz:

$$x < 3 * y + 2, \quad x = 1 \vdash x = y$$

2. Extrahiere arithmetische Formel für Widerspruchsbeweis

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

3. Setze Gleichheiten in andere Ungleichungen ein

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

SupInf: WIDERLEGUNGSBEISPIEL

1. Anfangssequenz:

$$x < 3 * y + 2, \quad x = 1 \vdash x = y$$

2. Extrahiere arithmetische Formel für Widerspruchsbeweis

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

3. Setze Gleichheiten in andere Ungleichungen ein

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

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

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

SupInf: WIDERLEGUNGSBEISPIEL

1. Anfangssequenz:

$$x < 3*y + 2, \quad x = 1 \vdash x = y$$

2. Extrahiere arithmetische Formel für Widerspruchsbeweis

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

3. Setze Gleichheiten in andere Ungleichungen ein

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

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

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

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

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

SupInf: WIDERLEGUNGSBEISPIEL

1. Anfangssequenz:

$$x < 3 * y + 2, \quad x = 1 \vdash x = y$$

2. Extrahiere arithmetische Formel für Widerspruchsbeweis

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

3. Setze Gleichheiten in andere Ungleichungen ein

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

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

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

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

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

6. Wende Sup-Inf Basismethode auf jedes Disjunkt an

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

SupInf: WIDERLEGUNGSBEISPIEL

1. Anfangssequenz:

$$x < 3 * y + 2, \quad x = 1 \vdash x = y$$

2. Extrahiere arithmetische Formel für Widerspruchsbeweis

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

3. Setze Gleichheiten in andere Ungleichungen ein

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

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. Wende Sup-Inf Basismethode auf jedes Disjunkt an

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

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

SupInf: WIDERLEGUNGSBEISPIEL

1. Anfangssequenz:

$$x < 3*y + 2, \quad x = 1 \vdash x = y$$

2. Extrahiere arithmetische Formel für Widerspruchsbeweis

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

3. Setze Gleichheiten in andere Ungleichungen ein

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

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

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

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

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

6. Wende Sup-Inf Basismethode auf jedes Disjunkt an

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

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

– Es gibt ein ganzzahliges Gegenbeispiel

- **Arithmetische Typen**

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

- **Arithmetische Typen**

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

SupInf: ALLGEMEINE ARBEITSWEISE

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

SupInf: ALLGEMEINE ARBEITSWEISE

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

SupInf: ALLGEMEINE ARBEITSWEISE

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

SupInf: ALLGEMEINE ARBEITSWEISE

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 sind Standardrepräsentationen von Polynomen

SupInf: ALLGEMEINE ARBEITSWEISE

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 sind Standardrepräsentationen von Polynomen
- 4. Ersetze nichtlineare Teilausdrücke durch Variablen**

SupInf: ALLGEMEINE ARBEITSWEISE

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 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

SupInf: ALLGEMEINE ARBEITSWEISE

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 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

SupInf': ERWEITERUNGEN ZU SupInf

Ergänze arithmetische Kontextinformation

SupInf': ERWEITERUNGEN ZU SupInf

Ergänze arithmetische Kontextinformation

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

SupInf': ERWEITERUNGEN ZU SupInf

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

SupInf': ERWEITERUNGEN ZU SupInf

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 $\mathbb{Z}$

SupInf': ERWEITERUNGEN ZU SupInf

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 $|l_1@l_2| = |l_1| + |l_2|$

SupInf': ERWEITERUNGEN ZU SupInf

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

SupInf': ERWEITERUNGEN ZU SupInf

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 $g(g(g(g(g(a)))))) = a$
- Intuitiver Beweis (gezieltes Einsetzen) 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 (gezieltes Einsetzen) 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 (gezieltes Einsetzen) 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

DIE ENTSCHIEDUNGSPROZEDUR Eq

Entscheide quantorenfreie Gleichheiten

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

DIE ENTSCHEIDUNGSPROZEDUR Eq

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

DIE ENTSCHEIDUNGSPROZEDUR Eq

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

DIE ENTSCHEIDUNGSPROZEDUR Eq

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 Inferenzregel**
 - Regel **equality** verweist auf Systemprozedur der Lisp-Ebene
 - Eingebettet in die Taktik **Auto**

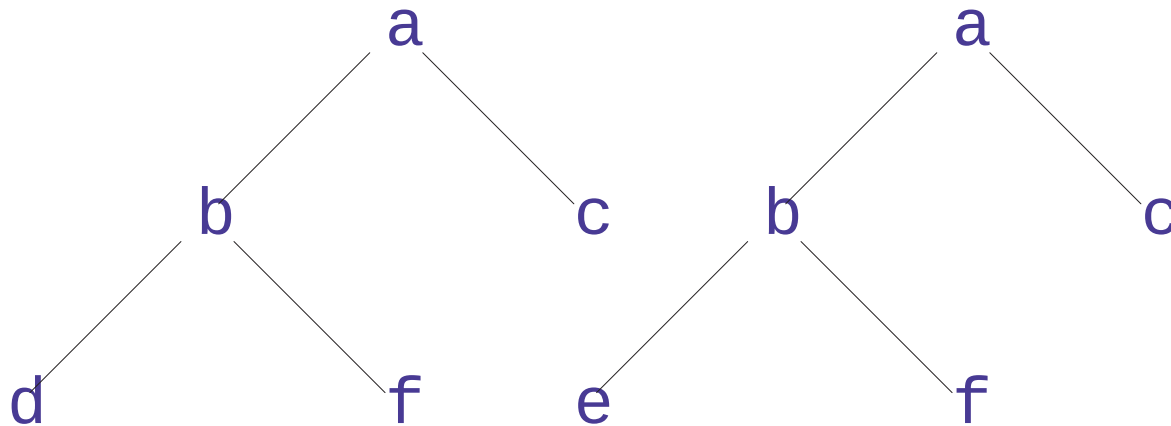
DIE ENTSCHEIDUNGSPROZEDUR Eq

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 Inferenzregel**
 - Regel **equality** verweist auf Systemprozedur der Lisp-Ebene
 - Eingebettet in die Taktik **Auto**

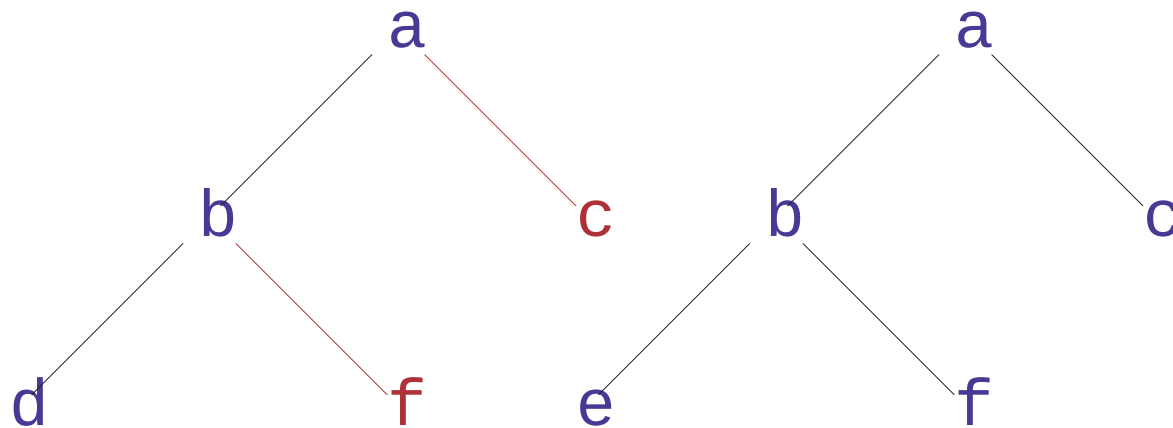
GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

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



GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

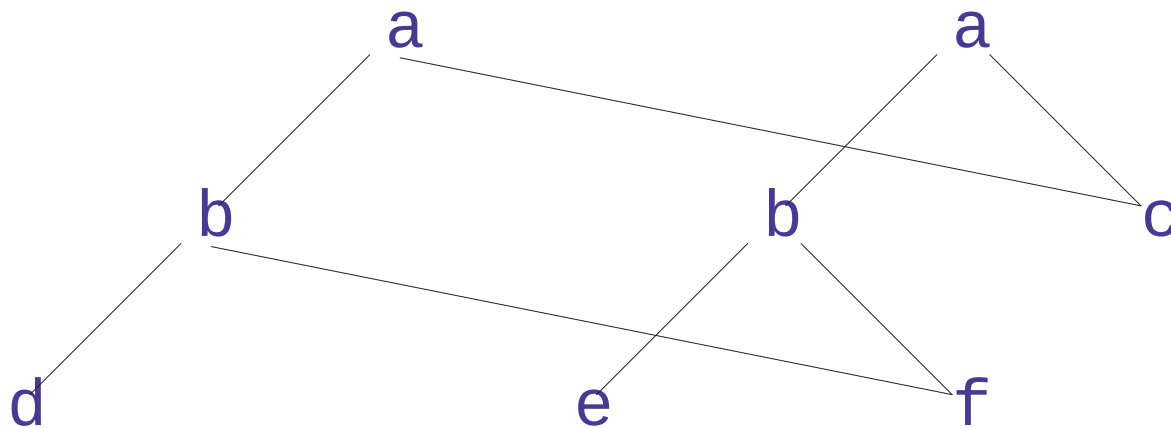
Zeige : $a(b(d, f), c) = a(b(e, f), c)$ **folgt aus** $d=e$



1. Verschmelze identische Knoten

GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

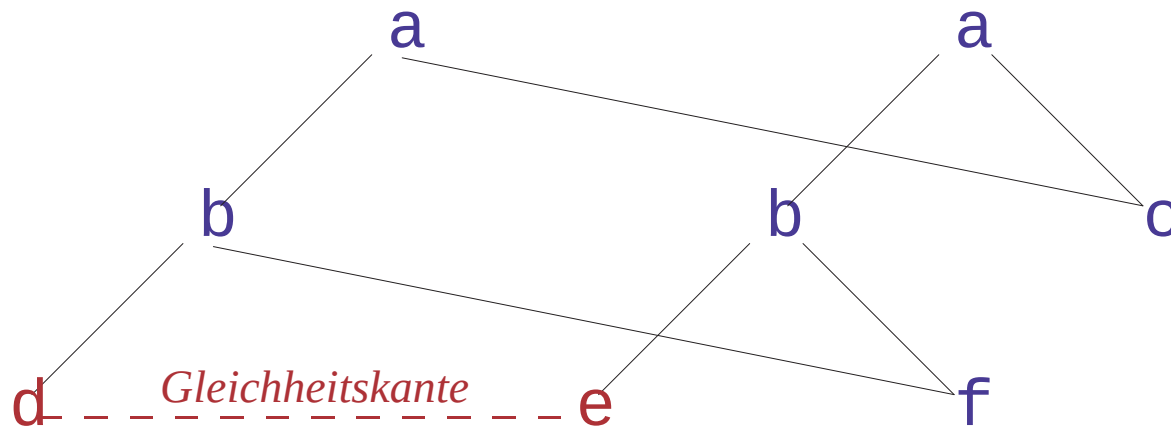
Zeige : $a(b(d, f), c) = a(b(e, f), c)$ **folgt aus** $d=e$



1. Verschmelze identische Knoten

GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

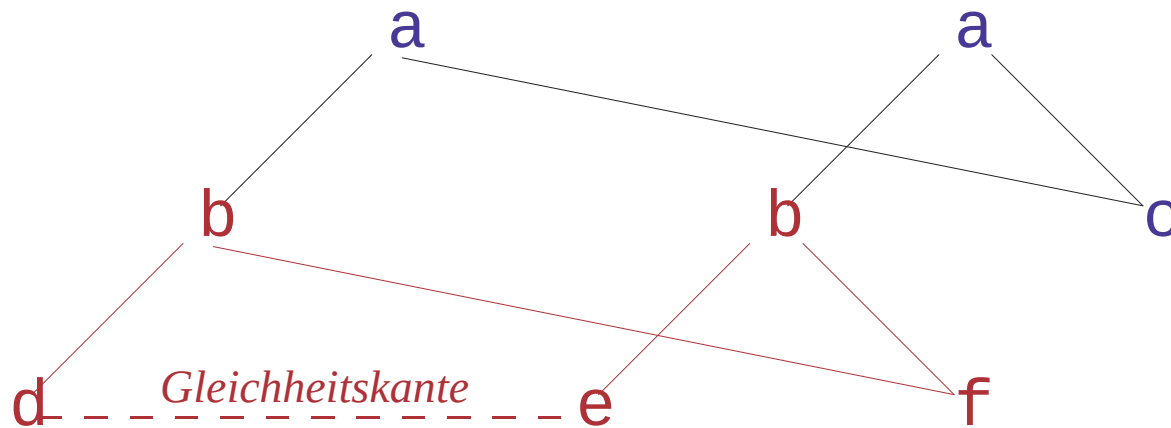
Zeige : $a(b(d, f), c) = a(b(e, f), c)$ **folgt aus** $d=e$



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

GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

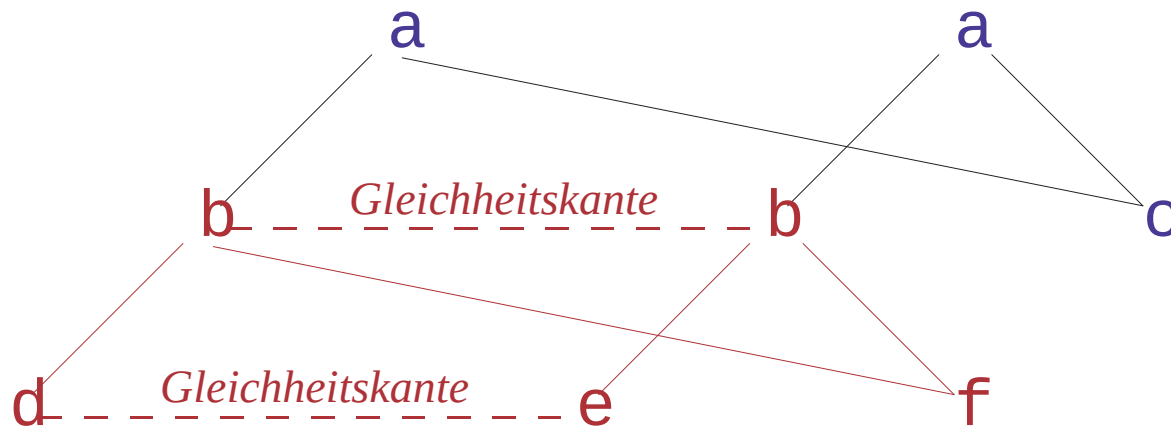
Zeige : $a(b(d, f), c) = a(b(e, f), c)$ **folgt aus** $d=e$



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

GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

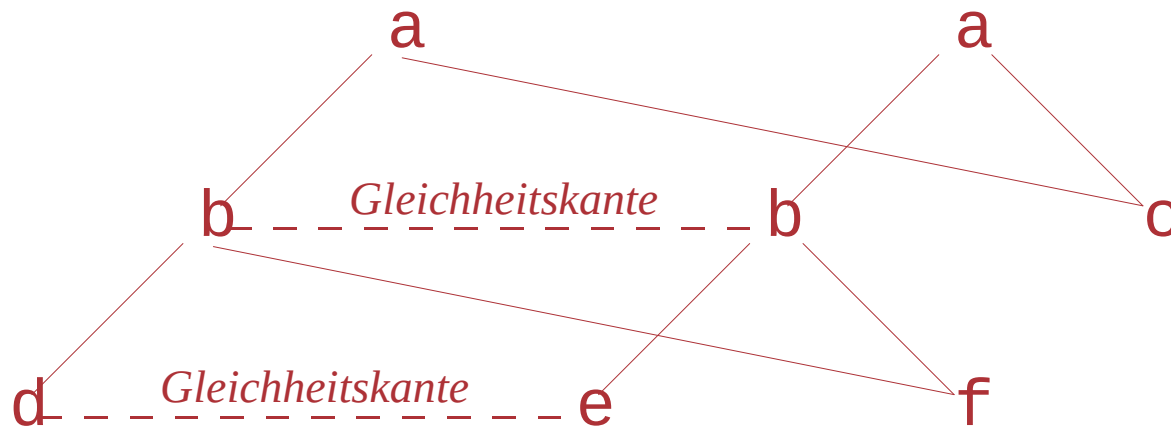
Zeige : $a(b(d, f), c) = a(b(e, f), c)$ **folgt aus** $d=e$



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

GLEICHHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

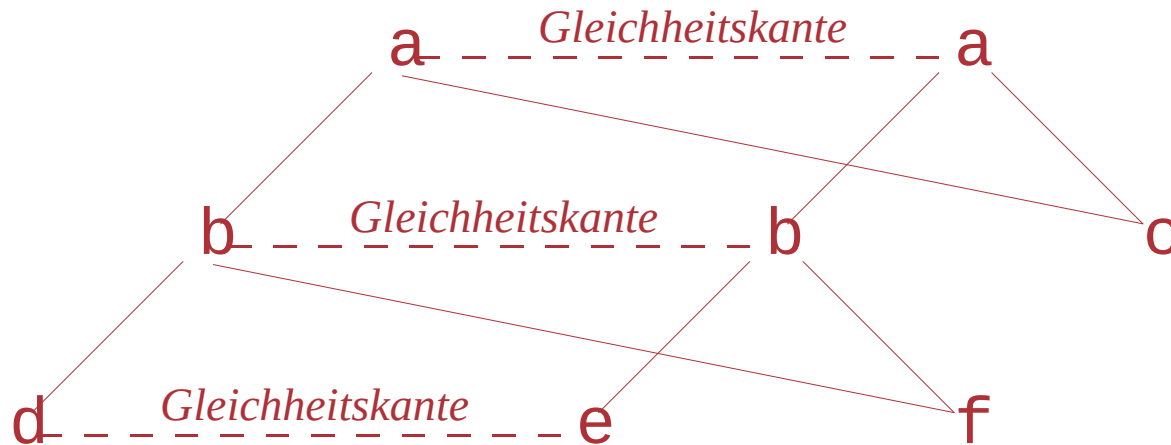
Zeige : $a(b(d, f), c) = a(b(e, f), c)$ **folgt aus** $d=e$



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

Eq: GRAPHENTHEORETISCHE KONZEPTE

- **Notationen für gerichteter Graphen $G = (V, E)$**
 - $l(v)$: 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

Eq: GRAPHENTHEORETISCHE KONZEPTE

- **Notationen für gerichteter Graphen $G = (V, E)$**
 - $l(v)$: 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
- **Begriffe für Äquivalenzrelationen 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: $u \sim_R v \Rightarrow (u, v) \in R$

Eq: GRAPHENTHEORETISCHE KONZEPTE

- **Notationen für gerichteter Graphen $G = (V, E)$**
 - $l(v)$: 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
- **Begriffe für Äquivalenzrelationen 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 $s_1=t_1, \dots, s_n=t_n$?

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 aller $s_i=t_i$ 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)

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 aller $s_i=t_i$ 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 aller $s_i=t_i$ 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 aller $s_i=t_i$ 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 $(\tau(s), \tau(t)) \in R^*$

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

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

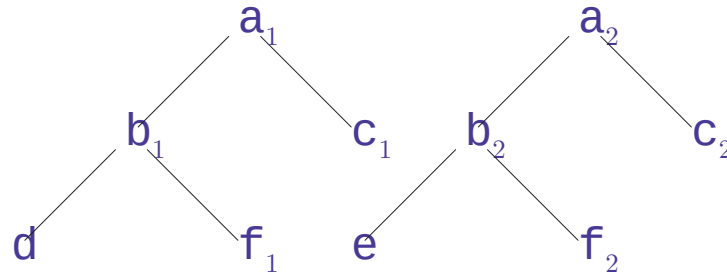
- **Vereinige** Äquivalenzklassen $[u]_R$ und $[v]_R$ in R

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

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

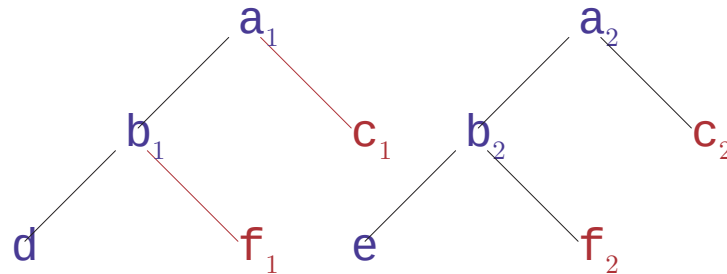
- Halte mit der modifizierten Relation** R **als Ergebnis**

KONGRUENZABSCHLUSS: $d = e \vdash a(b(d, f), c) = a(b(e, f), c)$



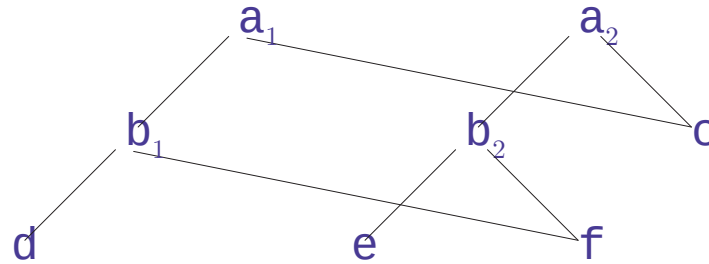
- **Graph ist Termbaum von $a(b(d, f), c)$ und $a(b(e, f), c)$**

KONGRUENZABSCHLUSS: $d = e \vdash a(b(d, f), c) = a(b(e, f), c)$



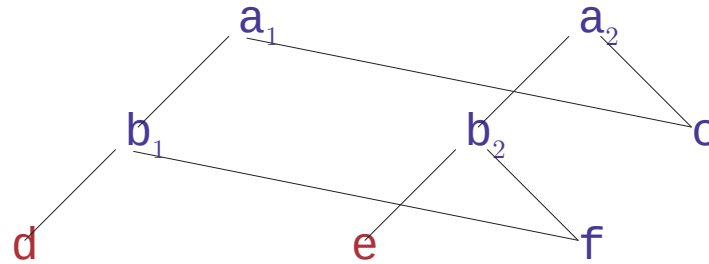
- **Graph ist Termbaum von $a(b(d, f), c)$ und $a(b(e, f), c)$**
 - Identische Teilausdrücke benutzen denselben Teilbaum

KONGRUENZABSCHLUSS: $d = e \vdash a(b(d, f), c) = a(b(e, f), c)$



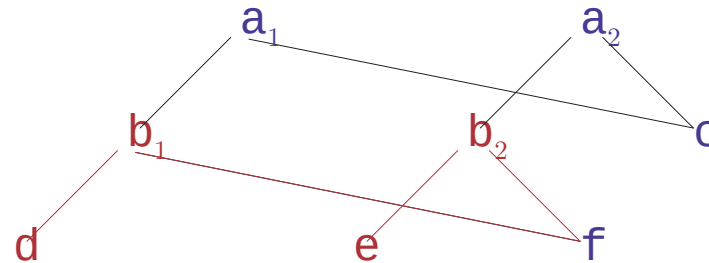
- **Graph ist Termbaum von $a(b(d, f), c)$ und $a(b(e, f), c)$**
 - Identische Teilausdrücke benutzen denselben Teilbaum
 - Initiale Relation: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\} \}$

KONGRUENZABSCHLUSS: $d = e \vdash a(b(d, f), c) = a(b(e, f), c)$



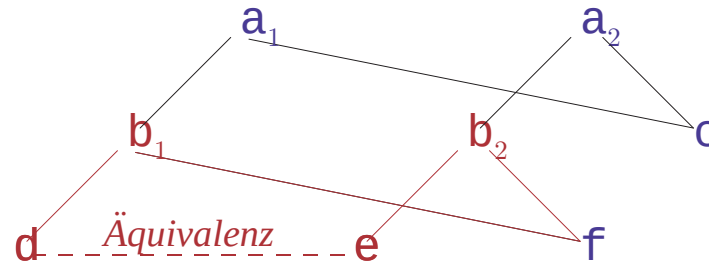
- **Graph ist Termbaum von $a(b(d, f), c)$ und $a(b(e, f), c)$**
 - Identische Teilausdrücke benutzen denselben Teilbaum
 - Initiale Relation: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\} \}$
- **Hinzunahme von $d = e$**

KONGRUENZABSCHLUSS: $d = e \vdash a(b(d, f), c) = a(b(e, f), c)$



- **Graph ist Termbaum von $a(b(d, f), c)$ und $a(b(e, f), c)$**
 - Identische Teilausdrücke benutzen denselben Teilbaum
 - Initiale Relation: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\} \}$
- **Hinzunahme von $d = e$**
Bestimme Vorgänger von $[d]_R (\{b_1\})$ und $[e]_R (\{b_2\})$

KONGRUENZABSCHLUSS: $d = e \vdash a(b(d, f), c) = a(b(e, f), c)$



- Graph ist **Termbaum** von $a(b(d, f), c)$ und $a(b(e, f), c)$

- Identische Teilausdrücke benutzen denselben Teilbaum

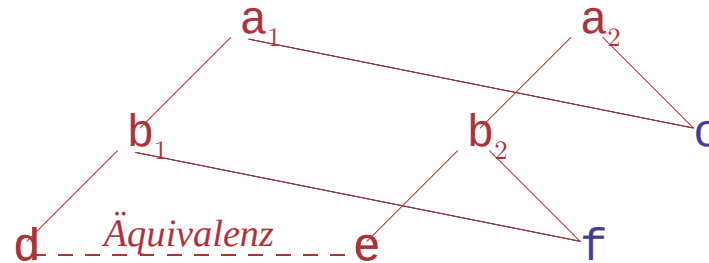
- Initiale Relation: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\} \}$

- **Hinzunahme von $d = e$**

Bestimme Vorgänger von $[d]_R (\{b_1\})$ und $[e]_R (\{b_2\})$

- Vereinige $[d]_R$ und $[e]_R$: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d, e\}, \{f\} \}$

KONGRUENZABSCHLUSS: $d = e \vdash a(b(d, f), c) = a(b(e, f), c)$



- Graph ist **Termbaum** von $a(b(d, f), c)$ und $a(b(e, f), c)$

- Identische Teilausdrücke benutzen denselben Teilbaum

- Initiale Relation: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\} \}$

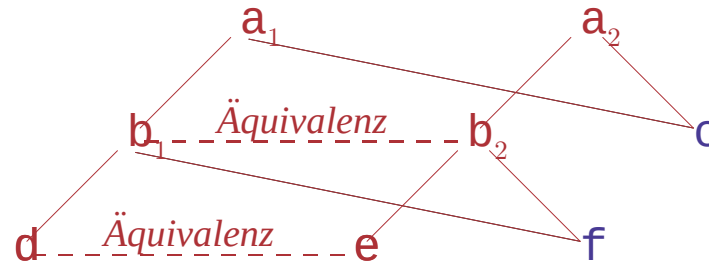
- **Hinzunahme von $d = e$**

Bestimme Vorgänger von $[d]_R (\{b_1\})$ und $[e]_R (\{b_2\})$

- Vereinige $[d]_R$ und $[e]_R$: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d, e\}, \{f\} \}$

Bestimme Vorgänger von $[b_1]_R (\{a_1\})$ und $[b_2]_R (\{a_2\})$

KONGRUENZABSCHLUSS: $d = e \vdash a(b(d, f), c) = a(b(e, f), c)$



- **Graph ist Termbaum von $a(b(d, f), c)$ und $a(b(e, f), c)$**

- Identische Teilausdrücke benutzen denselben Teilbaum

- Initiale Relation: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\} \}$

- **Hinzunahme von $d = e$**

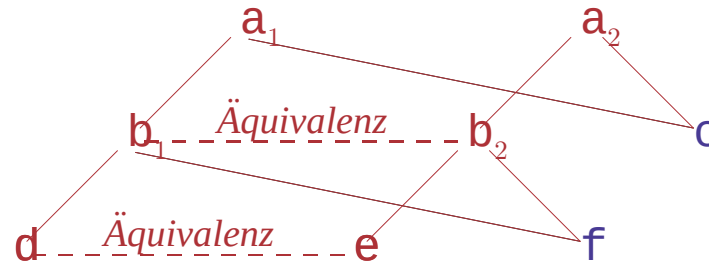
Bestimme Vorgänger von $[d]_R (\{b_1\})$ und $[e]_R (\{b_2\})$

- Vereinige $[d]_R$ und $[e]_R$: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d, e\}, \{f\} \}$

Bestimme Vorgänger von $[b_1]_R (\{a_1\})$ und $[b_2]_R (\{a_2\})$

- Vereinige $[b_1]_R$ und $[b_2]_R$: $R := \{ \{a_1\}, \{a_2\}, \{b_1, b_2\}, \{c\}, \{d, e\}, \{f\} \}$

KONGRUENZABSCHLUSS: $d = e \vdash a(b(d, f), c) = a(b(e, f), c)$



- **Graph ist Termbaum von $a(b(d, f), c)$ und $a(b(e, f), c)$**

- Identische Teilausdrücke benutzen denselben Teilbaum

- Initiale Relation: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\} \}$

- **Hinzunahme von $d = e$**

Bestimme Vorgänger von $[d]_R (\{b_1\})$ und $[e]_R (\{b_2\})$

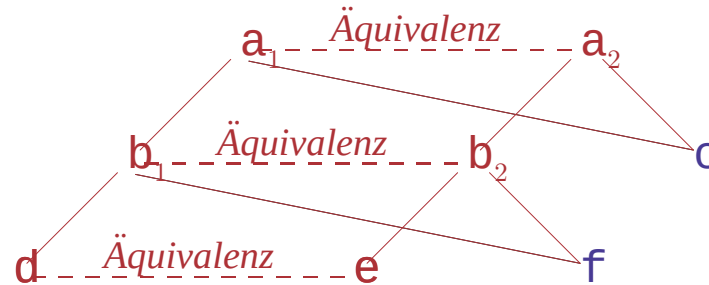
- Vereinige $[d]_R$ und $[e]_R$: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d, e\}, \{f\} \}$

Bestimme Vorgänger von $[b_1]_R (\{a_1\})$ und $[b_2]_R (\{a_2\})$

- Vereinige $[b_1]_R$ und $[b_2]_R$: $R := \{ \{a_1\}, \{a_2\}, \{b_1, b_2\}, \{c\}, \{d, e\}, \{f\} \}$

Bestimme Vorgänger von $[a_1]_R (\emptyset)$ und $[a_2]_R (\emptyset)$

KONGRUENZABSCHLUSS: $d = e \vdash a(b(d, f), c) = a(b(e, f), c)$



- **Graph ist Termbaum von $a(b(d, f), c)$ und $a(b(e, f), c)$**

- Identische Teilausdrücke benutzen denselben Teilbaum

- Initiale Relation: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\} \}$

- **Hinzunahme von $d = e$**

Bestimme Vorgänger von $[d]_R (\{b_1\})$ und $[e]_R (\{b_2\})$

- Vereinige $[d]_R$ und $[e]_R$: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d, e\}, \{f\} \}$

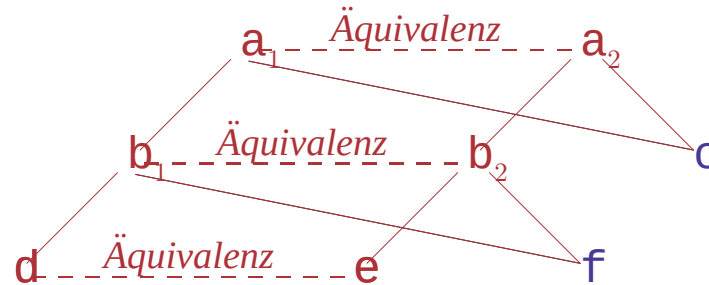
Bestimme Vorgänger von $[b_1]_R (\{a_1\})$ und $[b_2]_R (\{a_2\})$

- Vereinige $[b_1]_R$ und $[b_2]_R$: $R := \{ \{a_1\}, \{a_2\}, \{b_1, b_2\}, \{c\}, \{d, e\}, \{f\} \}$

Bestimme Vorgänger von $[a_1]_R (\emptyset)$ und $[a_2]_R (\emptyset)$

- Vereinige $[a_1]_R$ und $[a_2]_R$: $R := \{ \{a_1, a_2\}, \{b_1, b_2\}, \{c\}, \{d, e\}, \{f\} \}$

KONGRUENZABSCHLUSS: $d = e \vdash a(b(d, f), c) = a(b(e, f), c)$



- **Graph ist Termbaum von $a(b(d, f), c)$ und $a(b(e, f), c)$**

- Identische Teilausdrücke benutzen denselben Teilbaum

- Initiale Relation: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\} \}$

- **Hinzunahme von $d = e$**

Bestimme Vorgänger von $[d]_R (\{b_1\})$ und $[e]_R (\{b_2\})$

- Vereinige $[d]_R$ und $[e]_R$: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d, e\}, \{f\} \}$

Bestimme Vorgänger von $[b_1]_R (\{a_1\})$ und $[b_2]_R (\{a_2\})$

- Vereinige $[b_1]_R$ und $[b_2]_R$: $R := \{ \{a_1\}, \{a_2\}, \{b_1, b_2\}, \{c\}, \{d, e\}, \{f\} \}$

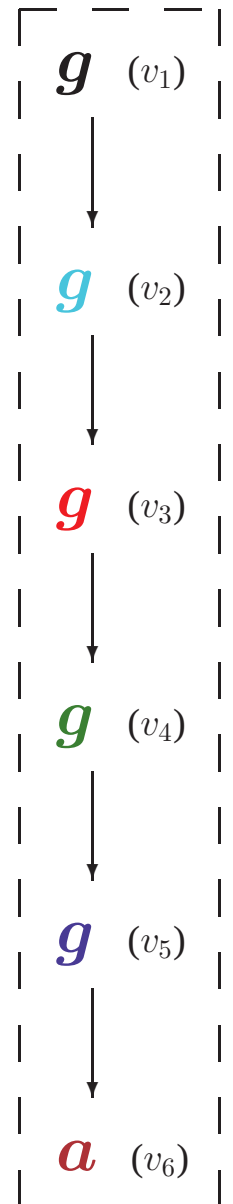
Bestimme Vorgänger von $[a_1]_R (\emptyset)$ und $[a_2]_R (\emptyset)$

- Vereinige $[a_1]_R$ und $[a_2]_R$: $R := \{ \{a_1, a_2\}, \{b_1, b_2\}, \{c\}, \{d, e\}, \{f\} \}$

Wurzelknoten der beiden Terme sind äquivalent

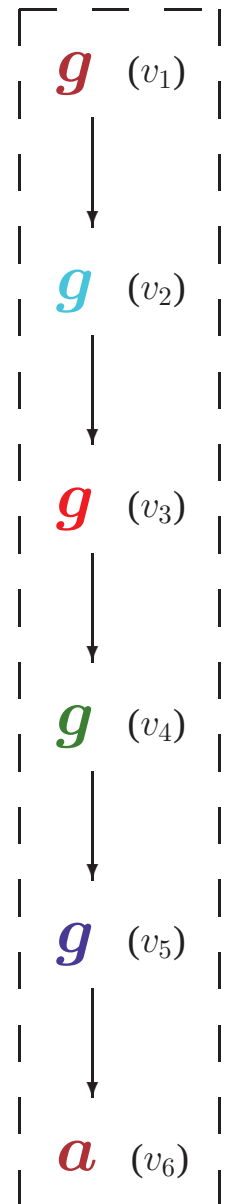
KONGRUENZABSCHLUSS: $g(g(g(a))) \doteq a, \quad g(g(g(g(g(a)))) \doteq a$

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



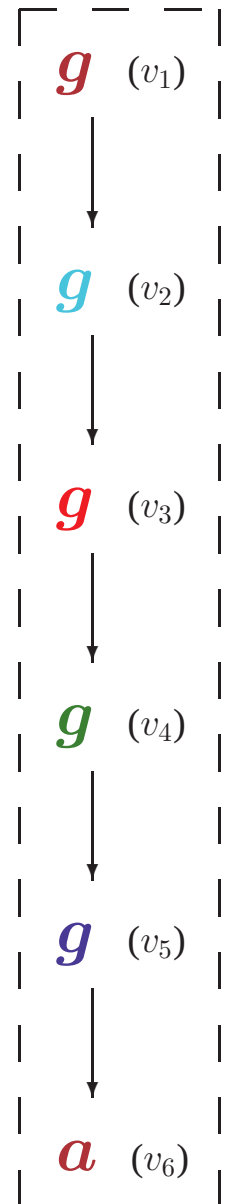
KONGRUENZABSCHLUSS: $g(g(g(a))) \doteq a$, $g(g(g(g(g(a)))) \doteq a$

- **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 $g(g(g(g(g(a)))) \doteq a$**
 - $R := \{ \{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\} \}$ ist abgeschlossen



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

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

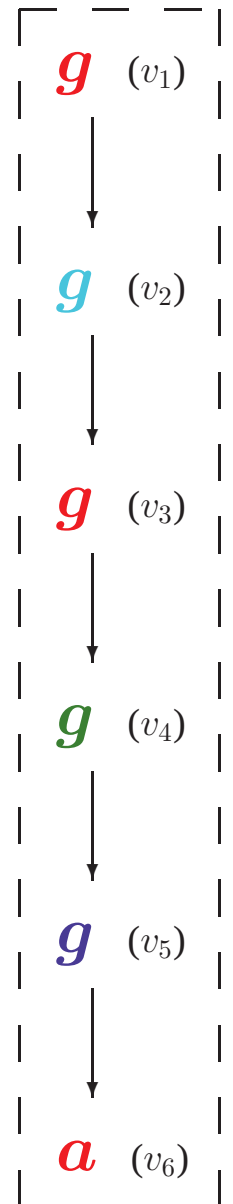


KONGRUENZABSCHLUSS: $g(g(g(a))) \dot{=} a, \quad g(g(g(g(g(a)))) \dot{=} a$

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

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: $g(g(g(a))) \doteq a, \quad g(g(g(g(g(a)))) \doteq a$

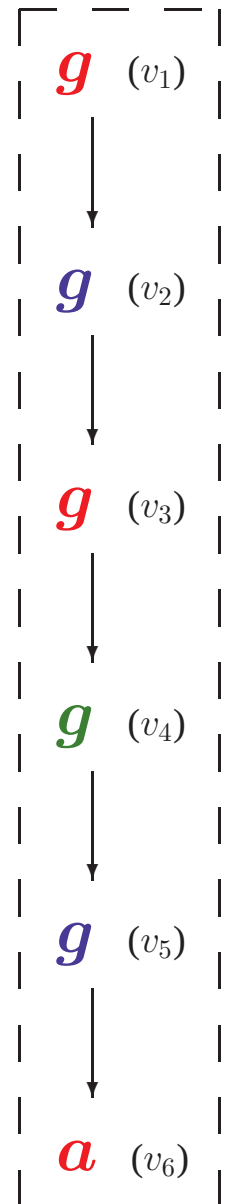
- **Graph ist Termbaum von $g(g(g(g(g(a))))$**
 - Initiale Relation: $R := \{ \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\} \}$
- **Hinzunahme von $g(g(g(g(g(a)))) \doteq a$**
 - $R := \{ \{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\} \}$ ist abgeschlossen
- **Hinzunahme von $g(g(g(a))) \doteq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

MERGE(R, v_3, v_6):

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

- $P_{v_2} := \{v_1\}, P_{v_5} := \{v_4\}, R := \{ \{v_1, v_6, v_3\}, \{v_4\}, \{v_2, v_5\} \}$
- Wegen $(v_2, v_5) \in R$ gilt $v_1 \sim_R v_4$ aber $[v_1]_R \neq [v_4]_R$



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

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

MERGE(R, v_3, v_6):

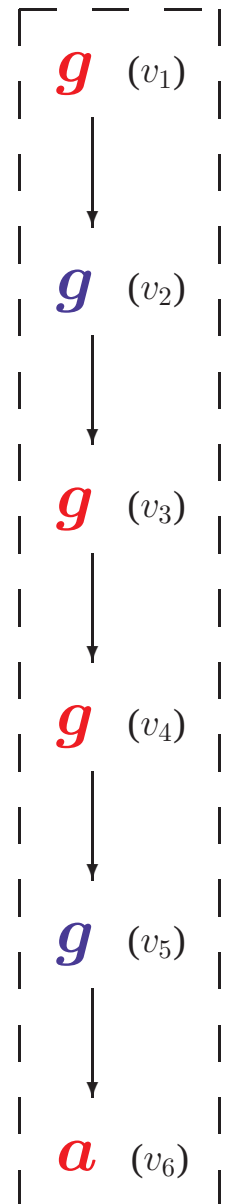
- $P_{v_3} := \{v_2\}, P_{v_6} := \{v_5\}, R := \{ \{v_1, v_6, v_3\}, \{v_2\}, \{v_4\}, \{v_5\} \}$
- Wegen $(v_3, v_6) \in R$ gilt $v_2 \sim_R v_5$ aber $[v_2]_R \neq [v_5]_R$

MERGE(R, v_2, v_5):

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

- $P_{v_1} := \{v_2, v_5\}, P_{v_4} := \{v_3\}, R := \{ \{v_1, v_6, v_3, v_4\}, \{v_2, v_5\} \}$
- Wegen $(v_6, v_4) \in R$ gilt $v_5 \sim_R v_3$ aber $[v_5]_R \neq [v_3]_R$



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

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

MERGE(R, v_3, v_6):

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

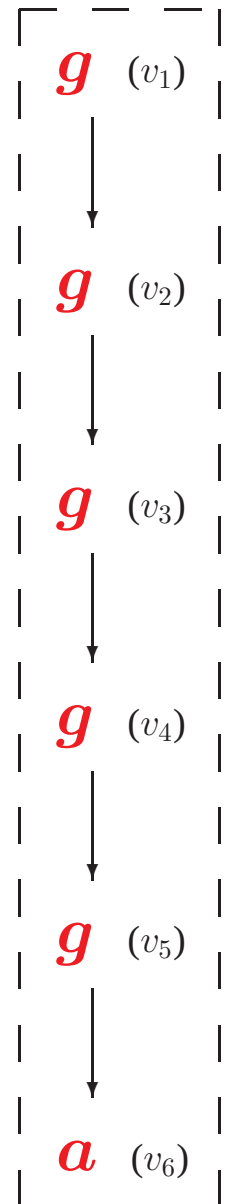
- $P_{v_2} := \{v_1\}, P_{v_5} := \{v_4\}, R := \{ \{v_1, v_6, v_3\}, \{v_4\}, \{v_2, v_5\} \}$
- Wegen $(v_2, v_5) \in R$ gilt $v_1 \sim_R v_4$ aber $[v_1]_R \neq [v_4]_R$

MERGE(R, v_1, v_4):

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

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



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

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

MERGE(R, v_3, v_6):

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

- $P_{v_2} := \{v_1\}, P_{v_5} := \{v_4\}, R := \{ \{v_1, v_6, v_3\}, \{v_4\}, \{v_2, v_5\} \}$
- Wegen $(v_2, v_5) \in R$ gilt $v_1 \sim_R v_4$ aber $[v_1]_R \neq [v_4]_R$

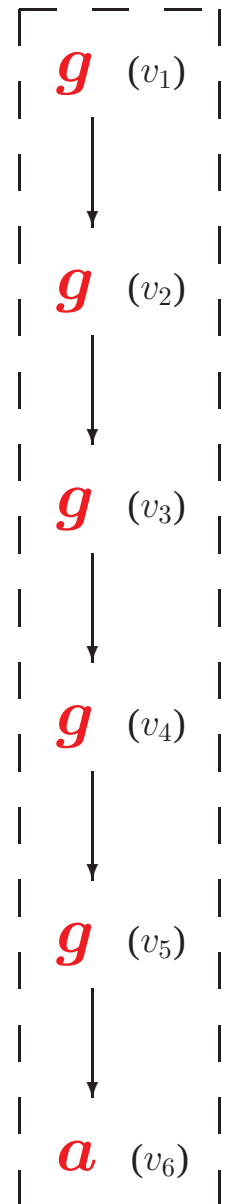
MERGE(R, v_1, v_4):

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

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

Alle Knoten sind äquivalent: $R=R^*$



GRENZEN VON ENTSCHEIDUNGSVERFAHREN

- **Weitere Theorien sind effektiv entscheidbar**
 - Schließen über Listenstrukturen
 - Geometrische Probleme
 - Aussagenlogik mit uninterpretierten Funktionssymbolen

GRENZEN VON ENTSCHEIDUNGSVERFAHREN

- **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

GRENZEN VON ENTSCHEIDUNGSVERFAHREN

- **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 unvorhersagbar