# Automatisierte Logik und Programmierung

## Einheit 14

## Entscheidungsprozeduren

1

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

# Entscheidungsprozeduren – algorithmische Inferenz

- **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 ∧ x>22 ∧ x<y ⇒ y=24` — viele Lemmata nötig

# Entscheidungsprozeduren – algorithmische Inferenz

- **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

# Entscheidungsprozeduren – algorithmische Inferenz

- **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!)
  – 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 Aussagen tauchen in vielen Erscheinungsformen auf

  $\texttt{x+1<y, 0<t} \vdash \texttt{(x+1)*t < y*t}$ entspricht $\texttt{x<y, 0<t} \vdash \texttt{x*t < y*t}$

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

  – 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

- **Anfangssequenz:** $\Gamma \vdash C_1 \lor \ldots \lor C_m$
  - $C_i$: 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

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

**1. Erzeuge Formeln für Widerspruchsbeweis:**

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

# Arith: Arbeitsweise am Beispiel

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

**1. Erzeuge Formeln für Widerspruchsbeweis:**

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

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

$$x+2y\geq z-1,\ z\geq i+5,\ i\geq 3x-2,\ x+2y\geq i,\ 3x\geq x+2y-1 \vdash ff$$

# Arith: Arbeitsweise am Beispiel

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

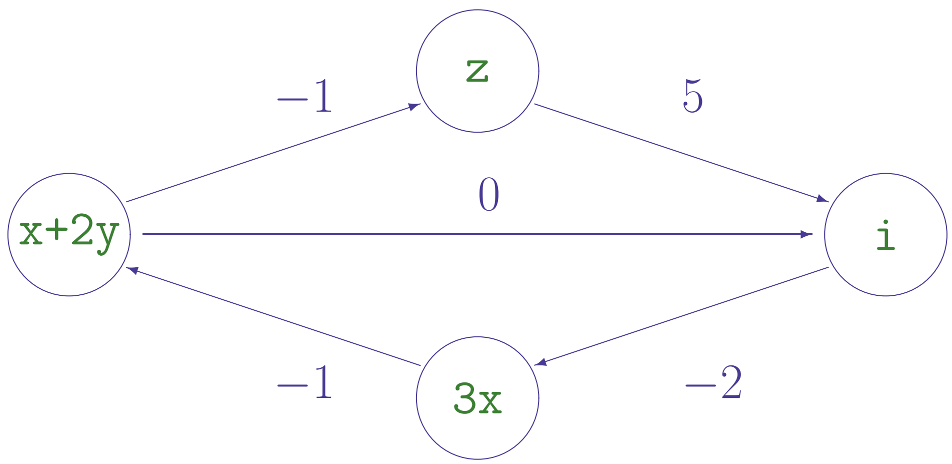
**1. Erzeuge Formeln für Widerspruchsbeweis:**

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

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

$$x+2y\geq z-1,\ \ z\geq i+5,\ \ i\geq 3x-2,\ \ x+2y\geq i,\ \ 3x\geq x+2y-1 \vdash ff$$

**3. Erzeuge Ordnungsgraphen:**



# Arith: Arbeitsweise am Beispiel

x+2y>z-2, i≤z-5, 3x<i+3, i≤2y+x ⊢ 2(y+x)-x ≥3x+2

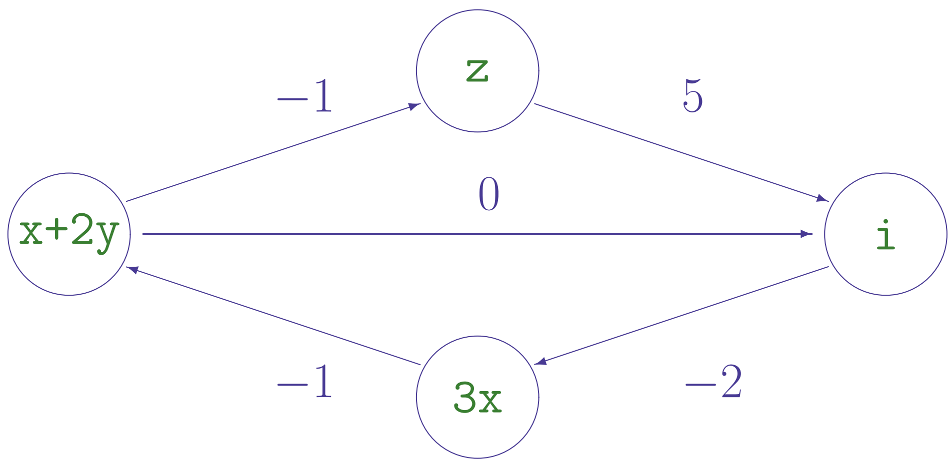
**1. Erzeuge Formeln für Widerspruchsbeweis:**

x+2y>z-2, i≤z-5, 3x<i+3, i≤2y+x, 2(y+x)-x < 3x+2 ⊢ ff

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

x+2y≥z-1, z≥i+5, i≥3x-2, x+2y≥i, 3x≥x+2y-1 ⊢ ff

**3. Erzeuge Ordnungsgraphen:**




**Ordnungsgraph hat positiven Zyklus … Sequenz ist gültig**

# Arith: Theoretische Fundament

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

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

- **Syntax: elementar-arithmetische Formeln** (ohne Induktion!)
  - Terme aufgebaut aus ganzzahligen Konstanten, Variablen und +, -, *
    Andersartige Terme gelten als unspezifierte Konstanten
  - Atomare Formeln: $t_1\,\rho\,t_2$, wobei $t_i$ Terme, $\rho \in \{<, \leq, >, \geq, =, \neq\}$
  - Formeln aufgebaut aus atomaren Formen mit $\neg$, $\wedge$, $\vee$ und $\Rightarrow$
  - Freie Variablen sind implizit all-quantifiziert

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

- **Syntax: elementar-arithmetische Formeln** (ohne Induktion!)
  - Terme aufgebaut aus ganzzahligen Konstanten, Variablen und +, -, *
    Andersartige Terme gelten als unspezifierte Konstanten
  - Atomare Formeln: $t_1\,\rho\,t_2$, wobei $t_i$ Terme, $\rho \in \{<, \leq, >, \geq, =, \neq\}$
  - Formeln aufgebaut aus atomaren Formen mit $\neg$, $\wedge$, $\vee$ und $\Rightarrow$
  - Freie Variablen sind implizit all-quantifiziert
- **Semantik charakterisiert durch Axiome** (Skript §4.3)
  1. Gleichheitsaxiome mit eingeschränkter Substitutivität

| | |
|---|---|
| $\texttt{x=x}$ | (Reflexivität) |
| $\texttt{x=y} \;\Rightarrow\; \texttt{y=x}$ | (Symmetrie) |
| $\texttt{x=y} \;\wedge\; \texttt{y=z} \;\Rightarrow\; \texttt{x=z}$ | (Transitivität) |
| $\texttt{x=y} \;\wedge\; \texttt{x}\,\rho\,\texttt{z} \;\Rightarrow\; \texttt{y}\,\rho\,\texttt{z}$ | (eingeschränkte Substitutivität) |
| $\texttt{x=y} \;\wedge\; \texttt{z}\,\rho\,\texttt{x} \;\Rightarrow\; \texttt{z}\,\rho\,\texttt{y}$ | |

Aus $\texttt{x=z} \;\wedge\; \texttt{x}\neq\texttt{x*y}$ folgt $\texttt{z}\neq\texttt{x*y}$, aber nicht $\texttt{z}\neq\texttt{z*y}$

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

- **Syntax: elementar-arithmetische Formeln** (ohne Induktion!)
  - Terme aufgebaut aus ganzzahligen Konstanten, Variablen und `+`, `-`, `*`
    Andersartige Terme gelten als unspezifierte Konstanten
  - Atomare Formeln: $t_1\,\rho\,t_2$, wobei $t_i$ Terme, $\rho \in \{<, \leq, >, \geq, =, \neq\}$
  - Formeln aufgebaut aus atomaren Formen mit $\neg$, $\wedge$, $\vee$ und $\Rightarrow$
  - Freie Variablen sind implizit all-quantifiziert

- **Semantik charakterisiert durch Axiome** (Skript §4.3)
  1. Gleichheitsaxiome mit eingeschränkter Substitutivität
  2. Ringaxiome der ganzen Zahlen

| | |
|---|---|
| `x+y=y+x`     `x*y=y*x` | (Kommutativgesetze) |
| `(x+y)+z=x+(y+z)`     `(x*y)*z=x*(y*z)` | (Assoziativgesetze) |
| `x*(y+z)=(x*z)+(y*z)` | (Distributivgesetz) |
| `x+0=x`     `x*1=x` | (Neutrale Elemente) |
| `x+(-x)=0` | (Inverses Element der Addition) |
| `x-y=x+(-y)` | (Definition der Subtraktion) |

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

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

| | |
|---|---|
| $\neg$(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) |

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

- **Syntax: elementar-arithmetische Formeln** (ohne Induktion!)
  - Terme aufgebaut aus ganzzahligen Konstanten, Variablen und +, -, *
    Andersartige Terme gelten als unspezifierte Konstanten
  - Atomare Formeln: $t_1\,\rho\,t_2$, wobei $t_i$ Terme, $\rho \in \{<, \leq, >, \geq, =, \neq\}$
  - Formeln aufgebaut aus atomaren Formen mit $\neg$, $\wedge$, $\vee$ und $\Rightarrow$
  - Freie Variablen sind implizit all-quantifiziert

- **Semantik charakterisiert durch Axiome** (Skript §4.3)
  1. Gleichheitsaxiome mit eingeschränkter Substitutivität
  2. Ringaxiome der ganzen Zahlen
  3. Axiome der diskreten linearen Ordnung
  4. Definitionsaxiome für Ordnungsrelationen und Ungleichheiten

$$x{\neq}y \;\Leftrightarrow\; \neg(x{=}y)$$
$$x{>}y \;\Leftrightarrow\; y{<}x$$
$$x{\leq}y \;\Leftrightarrow\; x{<}y \;\vee\; x{=}y$$
$$x{\geq}y \;\Leftrightarrow\; y{<}x \;\vee\; x{=}y$$

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

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

| | |
|---|---|
| $x\geq y \;\wedge\; z\geq w \;\Rightarrow\; x+z\geq y+w$ | (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) |

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

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

# Grundlagen einer effizienten Entscheidung

- **Entscheidbarkeit ermöglicht Widerspruchsbeweise**
  - $\Gamma \vdash C_1 \vee .. \vee C_n$ *ist gültig in $\mathcal{A}$, g.d.w.* $\Gamma, \neg C_1, \ldots, \neg C_n \vdash \mathsf{ff}$ *gültig*
- **Entscheidbarkeit erlaubt konjunktive 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**
  - $\Gamma = v_1 \geq u_1 + c_1, \ldots, v_n \geq u_n + c_n$ *ist genau dann widersprüchlich, wenn der Ordnungsgraph $\mathcal{G}$ von $\Gamma$ einen positiven Zyklus besitzt.*
  - Der Ordnungsgraph von $\Gamma$ 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 $\Gamma \vdash C_1 \vee .. \vee C_m$

**1. Transformiere Sequenz in Liste atomarer arithmetischer Formeln**
- Umwandlung in Widerspruchsbeweis $\Gamma, \neg C_1, .., \neg C_m \vdash \mathsf{ff}$
- 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 der Typentheorie leicht nachzuweisen**
  - Aussagen, die gültig in $\mathcal{A}$ sind, sind auch gültig in der Typentheorie
  - 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
    $\texttt{x}\geq\texttt{y} \wedge \texttt{i}\geq\texttt{j} \;\Rightarrow\; \texttt{x+i} \geq \texttt{y+j}$ mit Integerkonstanten `i` und `j`
  - Nichtriviale Monotonien müssen separat behandelt werden
    (Monotonie von `+`/`-` mit Variablen oder Monotonien mit `*`)
- **Zu schwach für lineare Ungleichungssysteme**
  - Monadische Polynome zerstören Bezüge zwischen Ungleichungen

# monotonicity: Behandlung von Monotonien

- **Arithmetische Komposition von Ungleichungen**
  – Erzeuge neue Hypothesen durch Anwendung von Monotonieaxiomen
  – z.B. Addition von $i \leq z-5$ und $3x<i+3$ ergibt $3x+i < i+3+z-5$
  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 $\texttt{i}\leq\texttt{z-5}$ und `3x<i+3` ergibt `3x+i < i+3+z-5`

    Multiplikation von $\texttt{x+y}>\texttt{z}$, $\texttt{2*x}\geq\texttt{0}$ ergibt $\texttt{2*x}^2\texttt{+2*x*y}\geq\texttt{2*x*z}$
- **`monotonicity`-Regel operiert auf Hypothesen**
  - z.B. `monotonicity` $i$+$j$ addiert Hypothesen $i$ und $j$

    Regel derzeit nur unvollständig im System!

# monotonicity: BEHANDLUNG VON MONOTONIEN

- **Arithmetische Komposition von Ungleichungen**
  - Erzeuge neue Hypothesen durch Anwendung von Monotonieaxiomen
  - z.B. Addition von $i \leq z-5$ und $3x<i+3$ ergibt $3x+i < i+3+z-5$

    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 $j$

    Regel derzeit nur unvollständig im System!
- **Effizient durch Verwendung von Monotonietabellen**
  - Tabellen beschreiben Kombination verschiedenartiger Ungleichungen

**Addition**

| | $z>w$ | $z\geq w$ | $z=w$ | $z\neq w$ |
|---|---|---|---|---|
| $x>y$ | $x+z\geq y+w+2$ | $x+z\geq y+w+1$ | $x+z\geq y+w+1$<br>$x+w\geq y+z+1$ | ----- |
| $x\geq y$ | $x+z\geq y+w+1$ | $x+z\geq y+w$ | $x+z\geq y+w$<br>$x+w\geq y+z$ | ----- |
| $x=y$ | $x+z\geq y+w+1$<br>$y+z\geq x+w+1$ | $x+z\geq y+w$<br>$y+z\geq x+w$ | $x+z=y+w$<br>$x+w=y+z$ | $x+z\neq y+w$<br>$x+w\neq y+z$ |
| $x\neq y$ | ----- | ----- | $x+z\neq y+w$<br>$x+w\neq y+z$ | ----- |

# monotonicity: Monotonietabellen

**Subtraktion**

| | z>w | z≥w | z=w | z≠w |
|---|---|---|---|---|
| x>y | x-w≥y-z+2 | x-w≥y-z+1 | x-w≥y-z+1<br>x-z≥y-w+1 | ----- |
| x≥y | x-w≥y-z+1 | x-w≥y-z | x-w≥y-z<br>x-z≥y-w | ----- |
| x=y | x-w≥y-z+1<br>y-w≥x-z+1 | x-w≥y-z<br>y-w≥x-z | x-w=y-z<br>y-w=x-z | x-w≠y-z<br>x-z≠y-w |
| x≠y | ----- | ----- | x-w≠y-z<br>x-z≠y-w | ----- |

**Multiplikation**

| | y≥z | y>z | y=z | y≠z |
|---|---|---|---|---|
| x>0 | x*y≥x*z | x*y>x*z | x*y=x*z | x*y≠x*z |
| x≥0 | xy*≥x*z | x*y≥x*z | x*y=x*z | ----- |
| x=0 | x*y=x*z<br>x*y=0 | x*y=x*z<br>x*y=0 | x*y=x*z<br>x*y=0 | x*y=x*z<br>x*y=0 |
| x≤0 | x*y≤x*z | x*y≤x*z | x*y=x*z | ----- |
| x<0 | x*y≤x*z | x*y<x*z | x*y=x*z | x*y≠x*z |
| x≠0 | ----- | x*y≠x*z | x*y=x*z | x*y≠x*z |

**Faktorisierung**

| | x*y>x*z | x*y≥x*z | x*y=x*z | x*y≠x*z |
|---|---|---|---|---|
| x>0 | y>z | y≥z | y=z | y≠z |
| x<0 | y<z | y≤z | y=z | y≠z |
| x≠0 | y≠z | ----- | y=z | y≠z |

# BEWEISE MIT `Arith` UND `monotonicity`

- **Beweis von** `x+y>z` $\wedge$ `2*x`$\geq$`z` $\Rightarrow$ `3*x+y`$\geq$`2*z-1`
  - Arith kann Beweis nicht ohne weitere Hilfe führen
  - Monotone Addition der beiden Ungleichungen in Annahme ergibt weitere Hypothese `x+y+2*x`$\geq$`z+z+1`
  - Beweis mit zusätzlicher Hypothese ist für `Arith` leicht zu führen

# BEWEISE MIT Arith UND monotonicity

- **Beweis von** $x+y>z \;\land\; 2*x\geq z \;\Rightarrow\; 3*x+y\geq 2*z-1$
  - Arith kann Beweis nicht ohne weitere Hilfe führen
  - Monotone Addition der beiden Ungleichungen in Annahme ergibt weitere Hypothese $x+y+2*x\geq z+z+1$
  - Beweis mit zusätzlicher Hypothese ist für `Arith` leicht zu führen
- **Beweis von** $x+2y>z-2 \;\land\; i\leq z-5 \;\land\; 3x<i+3 \;\Rightarrow\; y>x$
  - Addition der Hypothesen 1 und 3 ergibt $x+2y+i+3 \geq z-2+3x+1+1$
  - Subtraktion der Hypothesen 2 ergibt $x+2y+i+3-i \geq z-2+3x+1+1-(z-5)$
    Nach Normierung erhält man $2y \geq 2x+2$
  - Faktorisierung mit 2 liefert $y \geq x+1$
  - Beweis mit letzter Hypothese ist für `Arith` leicht zu führen

## Beweise mit Arith und monotonicity

- **Beweis von** $x+y>z \;\land\; 2*x\geq z \;\Rightarrow\; 3*x+y\geq 2*z-1$
  - Arith kann Beweis nicht ohne weitere Hilfe führen
  - Monotone Addition der beiden Ungleichungen in Annahme ergibt weitere Hypothese $x+y+2*x\geq z+z+1$
  - Beweis mit zusätzlicher Hypothese ist für `Arith` leicht zu führen
- **Beweis von** $x+2y>z-2 \;\land\; i\leq z-5 \;\land\; 3x<i+3 \;\Rightarrow\; y>x$
  - Addition der Hypothesen 1 und 3 ergibt $x+2y+i+3 \geq z-2+3x+1+1$
  - Subtraktion der Hypothesen 2 ergibt $x+2y+i+3-i \geq z-2+3x+1+1-(z-5)$ Nach Normierung erhält man $2y \geq 2x+2$
  - Faktorisierung mit 2 liefert $y \geq x+1$
  - Beweis mit letzter Hypothese ist für `Arith` leicht zu führen
- **Für praktische Anwendungen etwas zu mühsam**
  - Gleichungen sollten ohne Benutzerführung lösbar sein
  - Arithmetik linearer Ungleichungen benötigt komplexere Prozedur

# 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 $0 \le e_i$, 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 $\mathbb{Q}$

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

# SupInf: Arbeitsweise am Beispiel

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

# SupInf: Arbeitsweise am Beispiel

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

**2. Extrahiere arithmetische Formel für Widerspruchsbeweis**

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

# SupInf: Arbeitsweise am Beispiel

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

**2. Extrahiere arithmetische Formel für Widerspruchsbeweis**

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

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

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

# SupInf: Arbeitsweise am Beispiel

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

**2. Extrahiere arithmetische Formel für Widerspruchsbeweis**

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

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

$$\texttt{x+z+1}\leq\texttt{5} \land \texttt{y+8}\leq\texttt{3*z} \land \texttt{x+1}\leq\texttt{y} \land \texttt{2+z}\leq\texttt{2*y} \land \texttt{x}\leq\texttt{z-5}$$

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

$$\texttt{0}\leq\texttt{4-x-z} \land \texttt{0}\leq\texttt{3*z-y-8} \land \texttt{0}\leq\texttt{y-x-1} \land \texttt{0}\leq\texttt{2*y-z-2} \land \texttt{0}\leq\texttt{z-x-5}$$

# SupInf: Arbeitsweise am Beispiel

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

**2. Extrahiere arithmetische Formel für Widerspruchsbeweis**

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

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

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

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

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

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

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

# SupInf: Arbeitsweise am Beispiel

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

2. **Extrahiere arithmetische Formel für Widerspruchsbeweis**

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

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

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

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

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

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

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

6. $SUP(y)<INF(y)\ldots$

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

# SupInf: Widerlegungsbeispiel

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

# SupInf: Widerlegungsbeispiel

**1. Anfangssequenz:** `x<3*y+2, x=1 ⊢ x=y`

**2. Extrahiere arithmetische Formel für Widerspruchsbeweis**

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

# SupInf: Widerlegungsbeispiel

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

2. **Extrahiere arithmetische Formel für Widerspruchsbeweis**

   `x<3*y+2` $\wedge$ `x=1` $\wedge$ `x`$\neq$`y` $\vdash$ `ff`

3. **Setze Gleichheiten in andere Ungleichungen ein**

   `1<3*y+2` $\wedge$ `1`$\neq$`y` $\vdash$ `ff`

# SupInf: Widerlegungsbeispiel

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

**2. Extrahiere arithmetische Formel für Widerspruchsbeweis**

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

**3. Setze Gleichheiten in andere Ungleichungen ein**

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

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

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

# SupInf: Widerlegungsbeispiel

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

2. **Extrahiere arithmetische Formel für Widerspruchsbeweis**

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

3. **Setze Gleichheiten in andere Ungleichungen ein**

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

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

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

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

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

# SupInf: Widerlegungsbeispiel

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

2. **Extrahiere arithmetische Formel für Widerspruchsbeweis**

   `x<3*y+2` $\wedge$ `x=1` $\wedge$ `x`$\neq$`y` $\vdash$ `ff`

3. **Setze Gleichheiten in andere Ungleichungen ein**

   `1<3*y+2` $\wedge$ `1`$\neq$`y` $\vdash$ `ff`

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

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

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

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

6. **WendeSup-Inf Basismethode auf jedes Disjunkt an**

   **1.** `SUP(y)=`$\infty$ `INF(y)=2` **2.** `SUP(y)=0` `INF(y)=0`

# SupInf: Widerlegungsbeispiel

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

2. **Extrahiere arithmetische Formel für Widerspruchsbeweis**

   `x<3*y+2` $\wedge$ `x=1` $\wedge$ `x≠y` $\vdash$ `ff`

3. **Setze Gleichheiten in andere Ungleichungen ein**

   `1<3*y+2` $\wedge$ `1≠y` $\vdash$ `ff`

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

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

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

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

6. **WendeSup-Inf Basismethode auf jedes Disjunkt an**

   **1.** `SUP(y)=∞`   `INF(y)=2`   **2.** `SUP(y)=0`   `INF(y)=0`

7. $\{$`z`:$\mathbb{Z}$ | `SUP(y)≥z≥INF(y)`$\}$ **ist nicht leer …**

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

# SupInf: Formale Grundkonzepte

- **Arithmetische Typen**
  - $\mathbb{Z}$ (`int`), $\mathbb{Z}^{-0}$ (`int_nzero`)
  - $\mathbb{N}$ (`nat`), $\mathbb{N}^{+}$ (`nat_plus`)
  - $\{i\ldots\}$ (`int_upper`)
  - $\{i\ldots j^{-}\}$ (`int_seg`), $\{i\ldots j\}$ (`int_iseg`)

- **Arithmetische Literale**
  - $a=b \in T$ oder $a\neq b \in T$, wobei $T$ arithmetischer Typ
  - Arithmetische Ungleichungen mit $<$, $\leq$, $>$ und $\geq$
  - Negationen arithmetischer Literale

- **Arithmetische Formeln**
  - (Verschachtelte) Konjunktionen und Disjunktionen arithmetischer Literale

# SupInf: Allgemeine Arbeitsweise

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

1. **Extrahiere arithmetische Formel** $F = r_1 \wedge .. \wedge r_n \wedge \neg r_0$
   – 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

- **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?

- **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

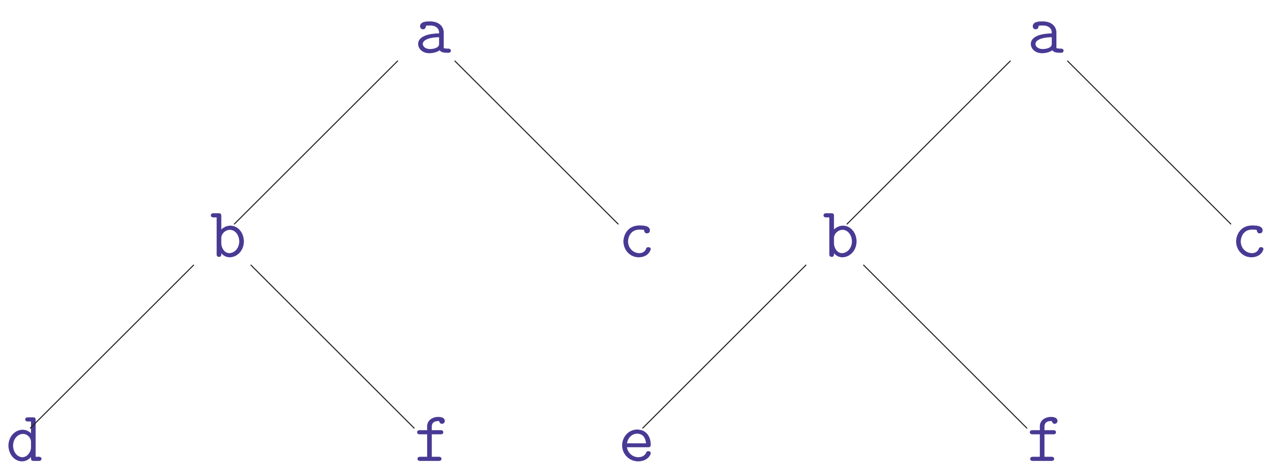
# Die Entscheidungsprozedur Eq

## Entscheide quantorenfreie Gleichheiten

- **Anfangssequenz:** $\Gamma, E_1, .., E_n \vdash E_0$
  – $E_i$ 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`

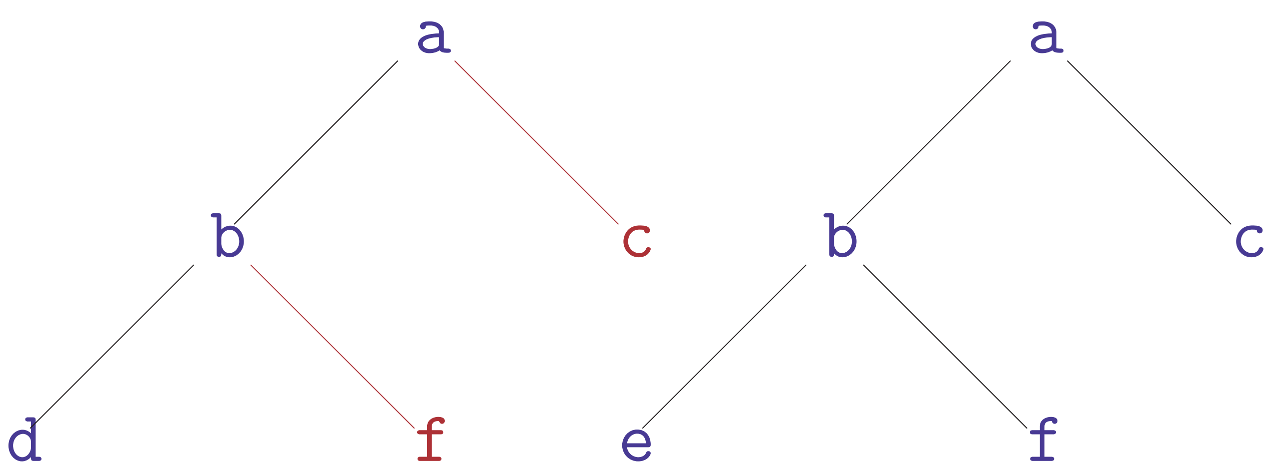
# Gleicheitsschliessen durch Kongruenzabschluss

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



# Gleicheitsschliessen durch Kongruenzabschluss

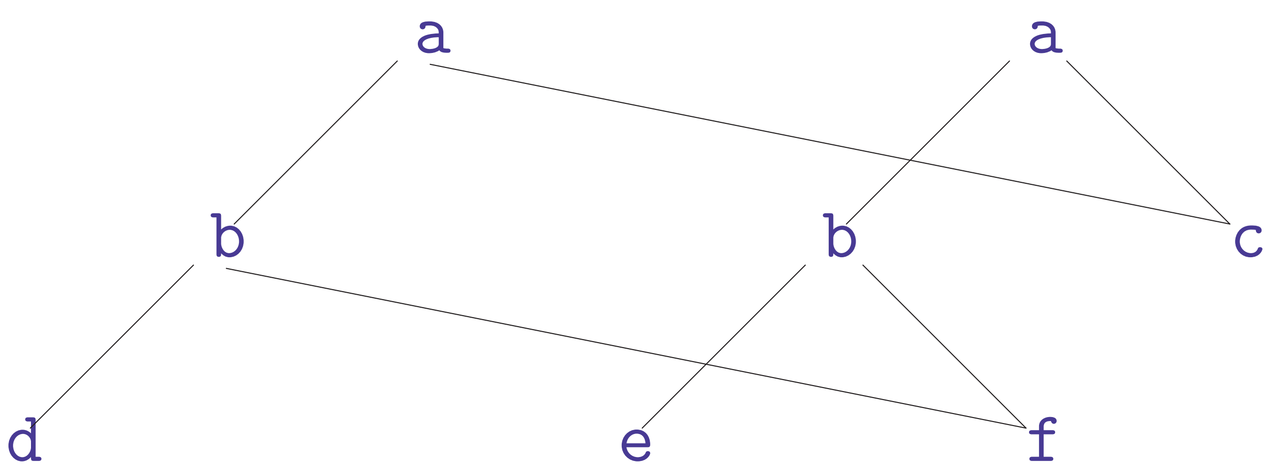
**Zeige :** `a(b(d,f),c) = a(b(e,f),c)` **folgt aus** `d=e`




1. Verschmelze identische Knoten

# Gleicheitsschliessen durch Kongruenzabschluss

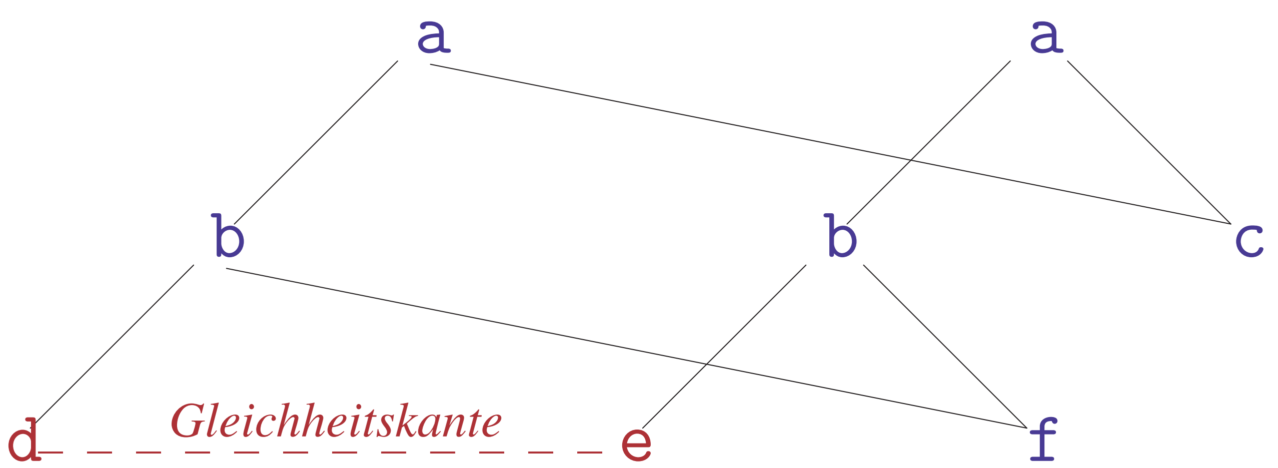
**Zeige :** `a(b(d,f),c) = a(b(e,f),c)` **folgt aus** `d=e`




1. Verschmelze identische Knoten

# Gleicheitsschliessen durch Kongruenzabschluss

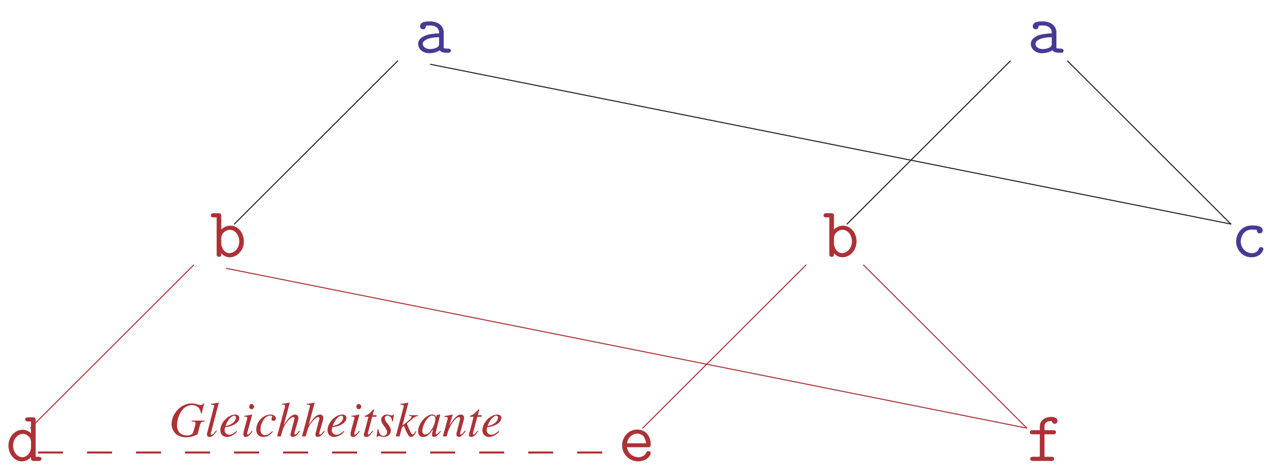
**Zeige :** `a(b(d,f),c) = a(b(e,f),c)` **folgt aus** `d=e`




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

# GLEICHEITSSCHLIESSEN DURCH KONGRUENZABSCHLUSS

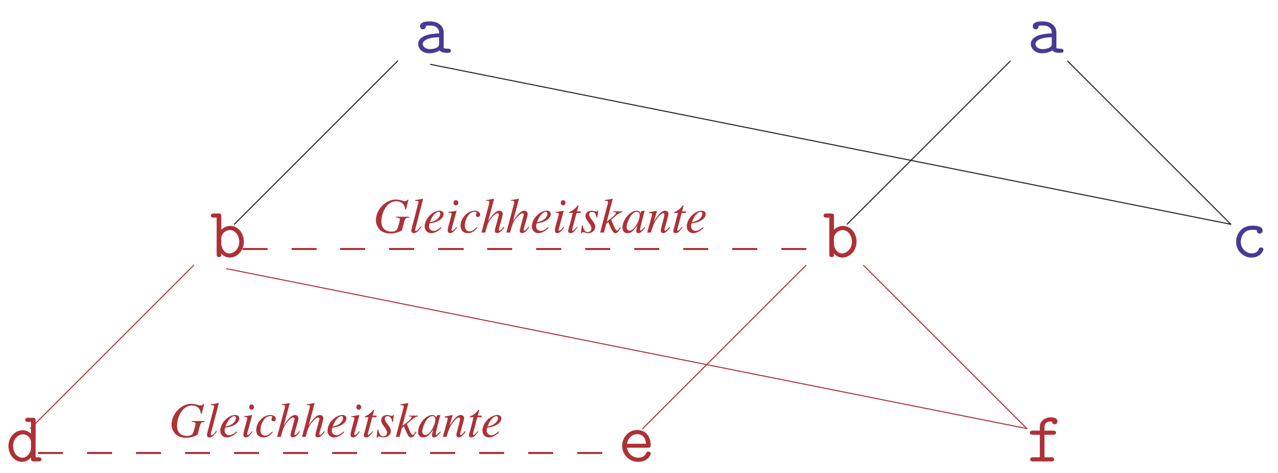
**Zeige :** `a(b(d,f),c) = a(b(e,f),c)` **folgt aus** `d=e`




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

# Gleichheitsschliessen durch Kongruenzabschluss

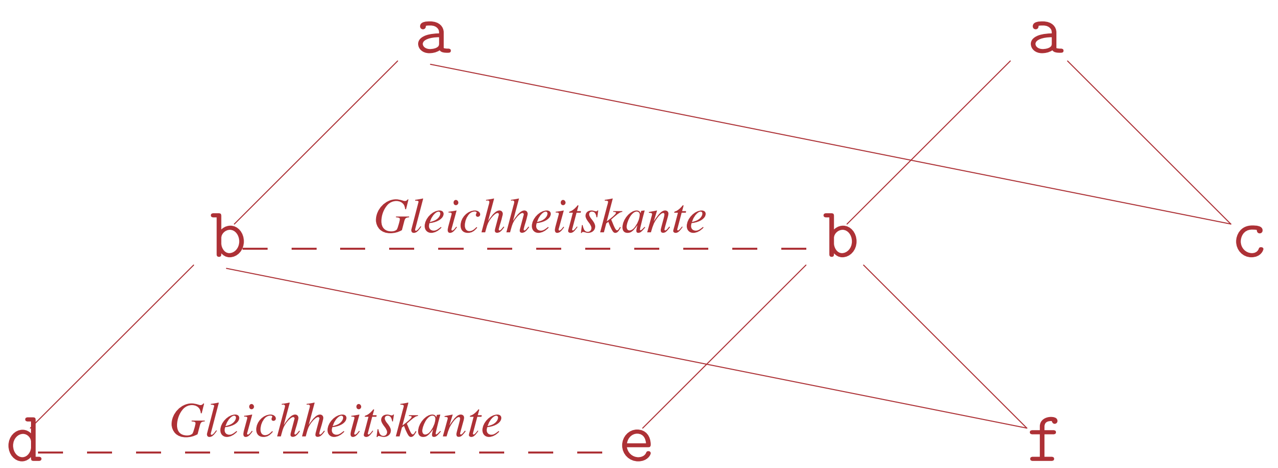
**Zeige :** `a(b(d,f),c) = a(b(e,f),c)` **folgt aus** `d=e`




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

# Gleichheitsschliessen durch Kongruenzabschluss

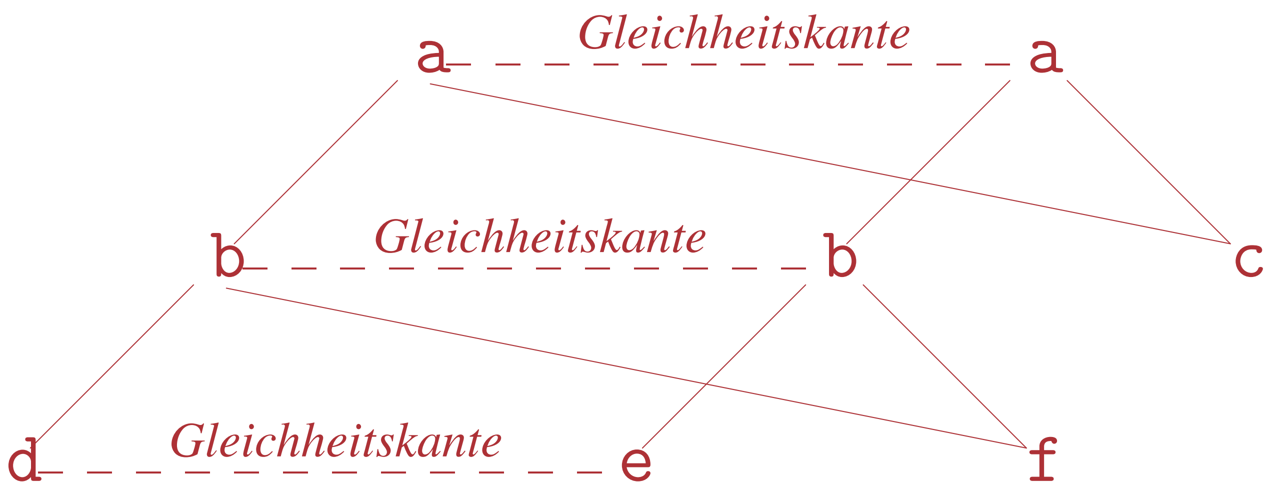
**Zeige :** `a(b(d,f),c) = a(b(e,f),c)` **folgt aus** `d=e`




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 $i$

- **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 $s_1{=}t_1,\ldots,s_n{=}t_n$?**

# Gleichheitsschliessen als Kongruenzabschluss

**Folgt $s = t$ aus $s_1$=$t_1$,...,$s_n$=$t_n$?**

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

# Gleichheitsschliessen als Kongruenzabschluss

**Folgt $s = t$ aus $s_1{=}t_1, \ldots, s_n{=}t_n$?**

- **Konstruiere Graph $G$ von $s, s_1, .., s_n,\ \ t, t_1, .., t_n$**
  - $G$ besteht aus Termbäumen von $s, s_1, .., s_n,\ \ t, t_1, .., t_n$
  - 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 $s_1{=}t_1,\ldots,s_n{=}t_n$?**

- **Konstruiere Graph $G$ von $s, s_1, .., s_n,\ t, t_1, .., t_n$**
  - $G$ besteht aus Termbäumen von $s, s_1, .., s_n,\ t, t_1, .., t_n$
  - 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 $s_1{=}t_1,\ldots,s_n{=}t_n$?**

- **Konstruiere Graph $G$ von $s, s_1, .., s_n,\ t, t_1, .., t_n$**
  - $G$ besteht aus Termbäumen von $s, s_1, .., s_n,\ t, t_1, .., t_n$
  - 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 $s_1{=}t_1,\ldots,s_n{=}t_n$?**

- **Konstruiere Graph $G$ von $s, s_1, .., s_n,\ t, t_1, .., t_n$**
  - $G$ besteht aus Termbäumen von $s, s_1, .., s_n,\ t, t_1, .., t_n$
  - 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 $R \cup \{(u,v)\}$

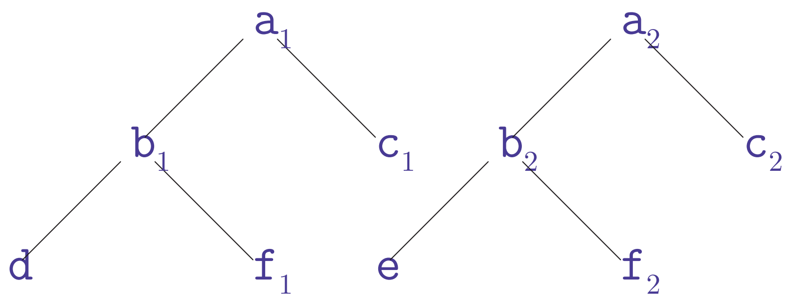
- **Algorithmus MERGE($R$,$u$,$v$)**
  – Eingabe: gerichteter Graph $G = (V, E)$, $u$, $v \in V$
  Äquivalenzrelation $R$ (abgeschlossen unter Kongruenzen)

- **Falls $u \sim_R v$, dann halte mit Ergebnis $R$**
  – Es gilt $(R \cup \{(u,v)\})^* = R$

- **Andernfalls modifiziere $R$ durch Verschmelzung**
  – Setze $P_u$ := $\{x \in V \mid \exists w \in [u]_R.\ x$ Vorgänger von $w\}$
  – Setze $P_v$ := $\{x \in V \mid \exists w \in [v]_R.\ x$ 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 $R$:=MERGE($R$,$x$,$y$)

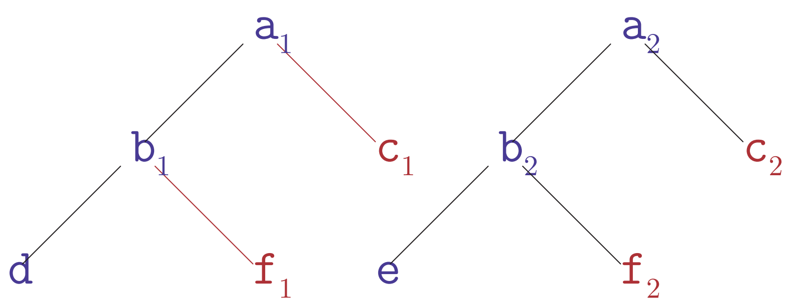
  **Halte mit der modifizierten Relation $R$ als Ergebnis**

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



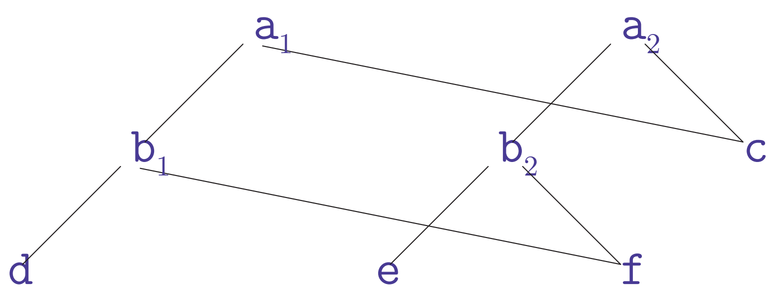

- **Graph ist Termbaum von $a(b(d, f), c)$ und $a(b(e, f), c)$**

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



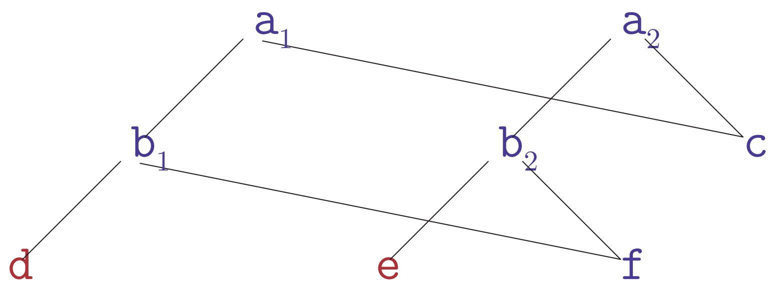

- **Graph ist Termbaum von $a(b(d,f),c)$ und $a(b(e,f),c)$**
  – Identische Teilausdrücke benutzen denselben Teilbaum

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



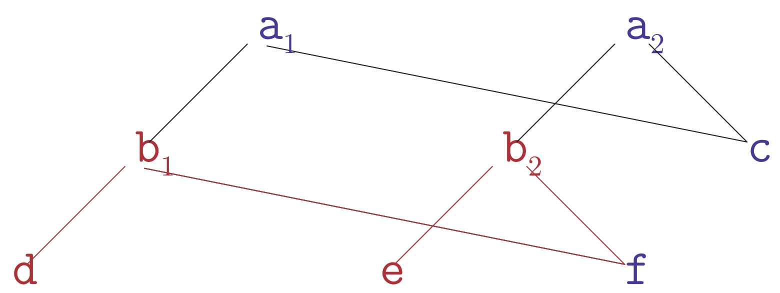

- **Graph ist Termbaum von $a(b(d,f),c)$ und $a(b(e,f),c)$**
  - Identische Teilausdrücke benutzen denselben Teilbaum
  - Initiale Relation: $R$ := $\{\,\{a_1\},\{a_2\},\{b_1\},\{b_2\},\{c\},\{d\},\{e\},\{f\}\,\}$

## 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 $d = e$**

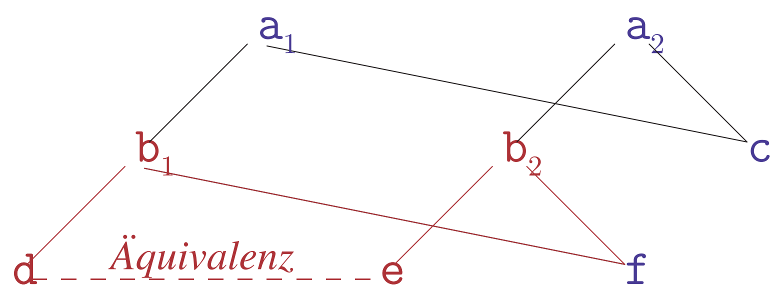
# KONGRUENZABSCHLUSS: $d = e \vdash a(b(d,f),c) = a(b(e,f),c)$




- **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 $[e]_R$ ($\{b_2\}$)

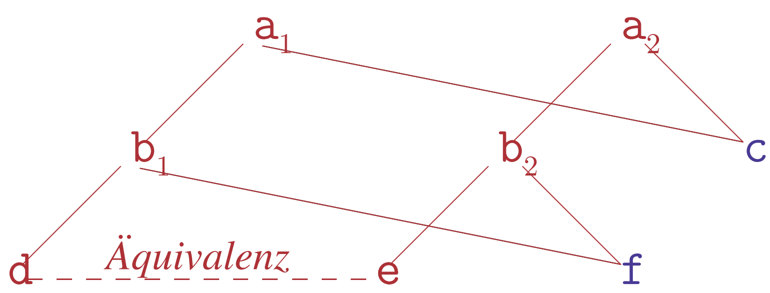
# KONGRUENZABSCHLUSS: $d = e \vdash a(b(d,f),c) = a(b(e,f),c)$




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

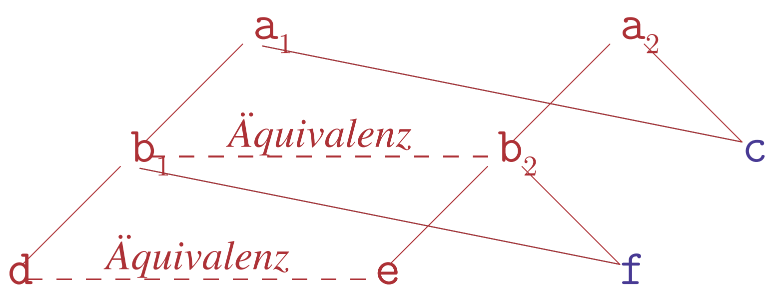



- **Graph ist Termbaum von $a(b(d,f),c)$ und $a(b(e,f),c)$**
  - Identische Teilausdrücke benutzen denselben Teilbaum
  - Initiale Relation: $R := \{ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\} \}$
- **Hinzunahme von $d = e$**

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



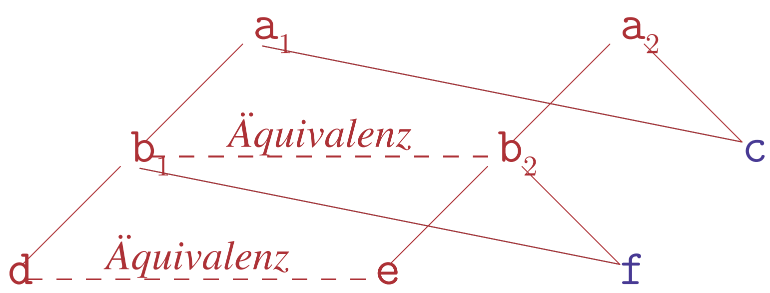

- **Graph ist Termbaum von $a(b(d,f),c)$ und $a(b(e,f),c)$**
  - Identische Teilausdrücke benutzen denselben Teilbaum
  - Initiale Relation: $R := \{\,\{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\}\,\}$

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

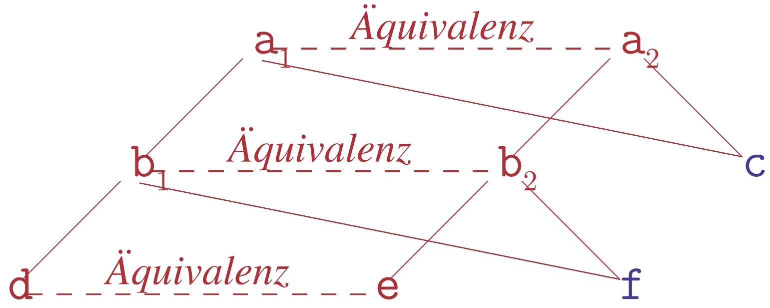
- **Hinzunahme von $d = e$**

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



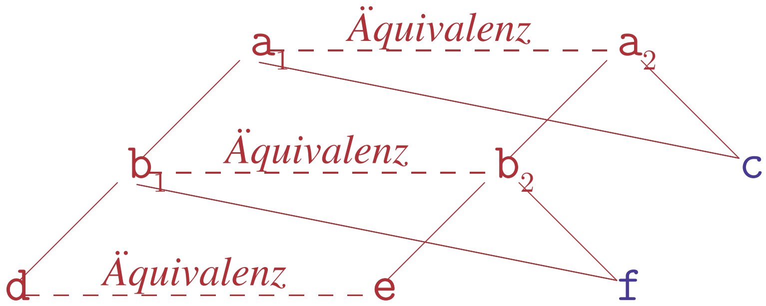

- **Graph ist Termbaum von $a(b(d,f),c)$ und $a(b(e,f),c)$**
  - Identische Teilausdrücke benutzen denselben Teilbaum
  - Initiale Relation: $R := \{\ \{a_1\}, \{a_2\}, \{b_1\}, \{b_2\}, \{c\}, \{d\}, \{e\}, \{f\}\ \}$
- **Hinzunahme von $d = e$**

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

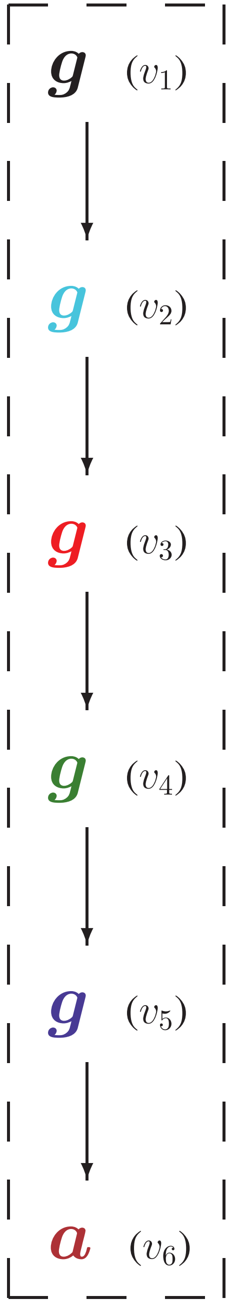
  Bestimme Vorgänger von $[b_1]_R$ ($\{a_1\}$) und $[b_2]_R$ ($\{a_2\}$)
  - Vereinige $[b_1]_R$ und $[b_2]_R$: $R := \{\, \{a_1\}, \{a_2\}, \{b_1, b_2\}, \{c\}, \{d,e\}, \{f\} \,\}$

  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**

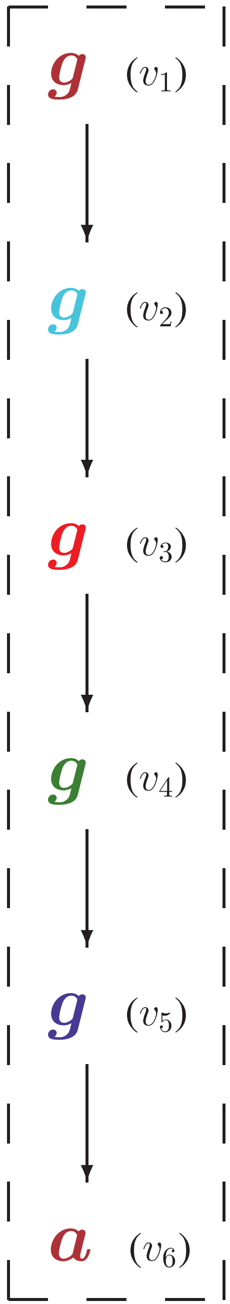
# Kongruenzabschluss: $g(g(g(a)))\doteq a, \quad g(g(g(g(g(a)))))\doteq a$

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



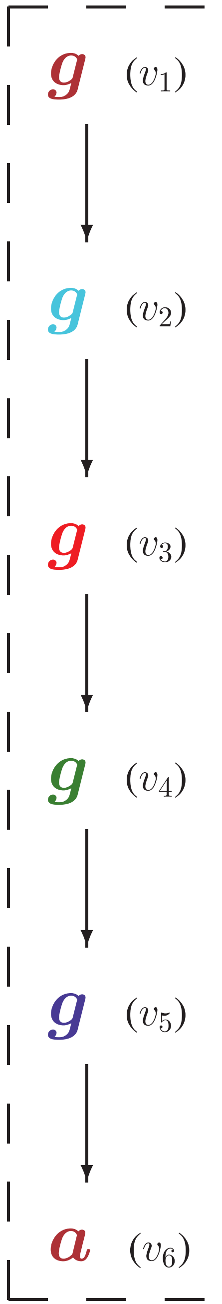
# Kongruenzabschluss: $g(g(g(a)))\doteq a, \quad g(g(g(g(g(a)))))\doteq a$

- **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



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

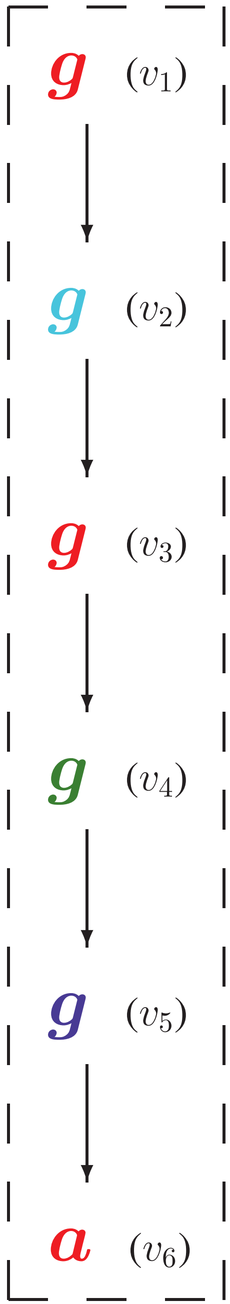
- **Graph ist Termbaum von $g(g(g(g(g(a)))))$**
  – Initiale Relation: $R := \{\, \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\}\,\}$
- **Hinzunahme von $g(g(g(g(g(a)))))\doteq a$**
  – $R := \{\, \{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}\,\}$ ist abgeschlossen
- **Hinzunahme von $g(g(g(a)))\doteq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



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

- **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 $[v_2]_R \neq [v_5]_R$



## 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 $[v_1]_R \neq [v_4]_R$

$$\begin{tikzcd}
g \; (v_1) \arrow[d] \\
g \; (v_2) \arrow[d] \\
g \; (v_3) \arrow[d] \\
g \; (v_4) \arrow[d] \\
g \; (v_5) \arrow[d] \\
a \; (v_6)
\end{tikzcd}$$

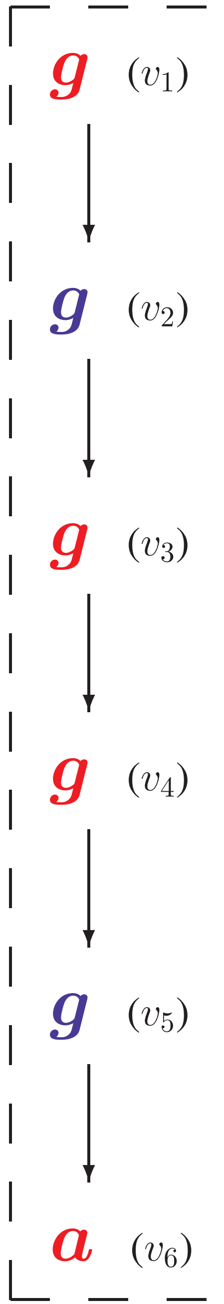
# KONGRUENZABSCHLUSS: $g(g(g(a)))\doteq a, \quad g(g(g(g(g(a)))))\doteq a$

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

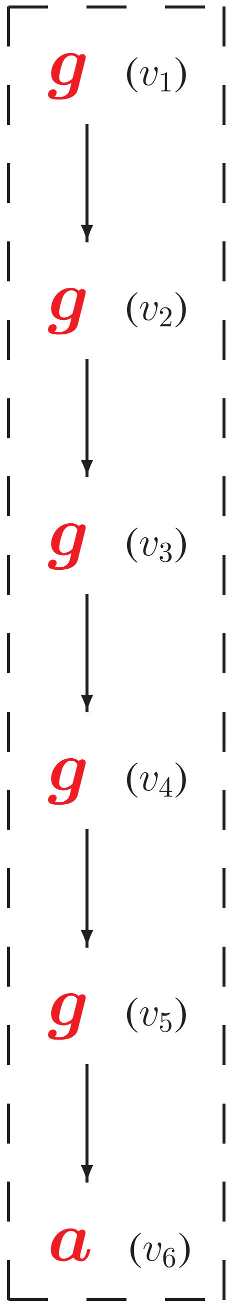
- **Graph ist Termbaum von $g(g(g(g(g(a)))))$**
  – Initiale Relation: $R := \{\, \{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}, \{v_6\} \,\}$
- **Hinzunahme von $g(g(g(g(g(a)))))\doteq a$**
  – $R := \{\, \{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\} \,\}$ ist abgeschlossen
- **Hinzunahme von $g(g(g(a)))\doteq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

  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 $g(g(g(a)))\doteq a$**

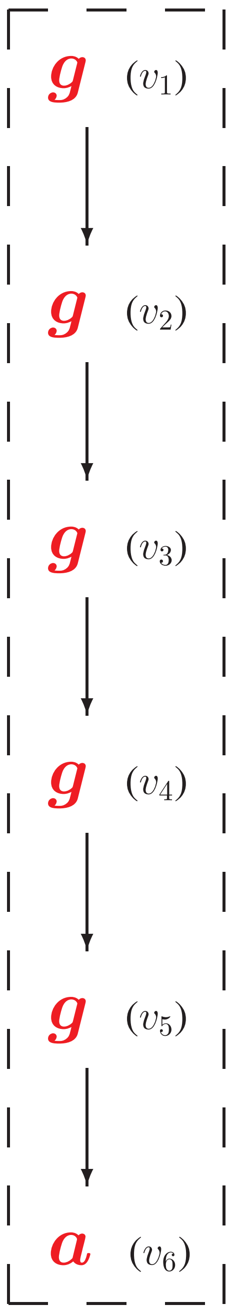
  MERGE($R$,$v_3$,$v_6$):
  - $P_{v_3} := \{v_2\}, \;\; P_{v_6} := \{v_5\}, \;\; R := \{\, \{v_1, v_6, v_3\}, \{v_2\}, \{v_4\}, \{v_5\}\, \}$
  - Wegen $(v_3, v_6) \in R$ gilt $v_2 \sim_R v_5$ aber $[v_2]_R \neq [v_5]_R$

  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
- **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

# 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

# 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