

Inferenzmethoden



Einheit 18

Offene Themen



1. Effizientere allgemeine Beweissuche
 2. Erweiterung auf andere Logiken
 3. Höhere Formen des Schließens
 4. Anwendungen
-
5. Ausblick

VERDICHUNG DER ALLGEMEINEN BEWEISSUCHE

- **Spezialanalyse ersetzt explizite Beweisschritte**
 - Verdichtete Form liefert dieselbe Information wie ausführlicher Beweis

VERDICHTUNG DER ALLGEMEINEN BEWEISSUCHE

- **Spezialanalyse ersetzt explizite Beweisschritte**
 - Verdichtete Form liefert dieselbe Information wie ausführlicher Beweis
- **Viele Beweise brauchen Klauselinstanzen**
 - Explizite Kopien der Klausel vergrößern Matrix und Suchraum
 - Alternative: Dynamische Aufspaltung oder Faktorisierung von Klauseln

VERDICHTUNG DER ALLGEMEINEN BEWEISSUCHE

- **Spezialanalyse ersetzt explizite Beweisschritte**
 - Verdichtete Form liefert dieselbe Information wie ausführlicher Beweis
- **Viele Beweise brauchen Klauselinstanzen**
 - Explizite Kopien der Klausel vergrößern Matrix und Suchraum
 - Alternative: Dynamische Aufspaltung oder Faktorisierung von Klauseln
- **Manche Beweise durchlaufen Zyklen**
 - Eine Kette von Argumenten wird mehrfach benutzt
 - Ähnlich zu Schleifen in Programmen muß ein “Ausstieg” vorhanden sein
 - Vermeide Kopien durch Bestimmung der “Zyklusmultiplizität”
 - Nichtterminierende (“tautologische”) Zyklen sind für Beweise unbrauchbar

VERDICHTUNG DER ALLGEMEINEN BEWEISSUCHE

- **Spezialanalyse ersetzt explizite Beweisschritte**
 - Verdichtete Form liefert dieselbe Information wie ausführlicher Beweis
- **Viele Beweise brauchen Klauselinstanzen**
 - Explizite Kopien der Klausel vergrößern Matrix und Suchraum
 - Alternative: Dynamische Aufspaltung oder Faktorisierung von Klauseln
- **Manche Beweise durchlaufen Zyklen**
 - Eine Kette von Argumenten wird mehrfach benutzt
 - Ähnlich zu Schleifen in Programmen muß ein “Ausstieg” vorhanden sein
 - Vermeide Kopien durch Bestimmung der “Zyklusmultiplizität”
 - Nichtterminierende (“tautologische”) Zyklen sind für Beweise unbrauchbar
- **Viele Anwendungen haben große Faktenmengen**
 - Ausgangsmatrix zu groß für effiziente Suche mit allgemeinen Verfahren
 - Schneller Zugriff erfordert Datenbanktechniken und Indizierung

PARALLELITÄT: AUSNUTZUNG BESSERER HARDWARE

- **Leistungssteigerung durch Verdichtung hat Grenzen**
 - Es gibt kaum syntaktische Kriterien zur Auswahl der besten Strategie
 - Aussagenlogisches Beweisen ist bereits \mathcal{NP} -vollständig
 - Kann man Nichtdeterminismus ausnutzen?

PARALLELITÄT: AUSNUTZUNG BESSERER HARDWARE

- **Leistungssteigerung durch Verdichtung hat Grenzen**
 - Es gibt kaum syntaktische Kriterien zur Auswahl der besten Strategie
 - Aussagenlogisches Beweisen ist bereits \mathcal{NP} -vollständig
 - Kann man Nichtdeterminismus ausnutzen?
- **Verteile alternative Wege auf mehrere Prozessoren**
 - Verarbeite Alternativen (Konnektionsmengen, Substitutionen, ...) simultan
 - Lasse alternative Beweiser parallel laufen
 - Die erste korrekte Lösung reicht als Beweis

PARALLELITÄT: AUSNUTZUNG BESSERER HARDWARE

- **Leistungssteigerung durch Verdichtung hat Grenzen**
 - Es gibt kaum syntaktische Kriterien zur Auswahl der besten Strategie
 - Aussagenlogisches Beweisen ist bereits \mathcal{NP} -vollständig
 - Kann man Nichtdeterminismus ausnutzen?
- **Verteile alternative Wege auf mehrere Prozessoren**
 - Verarbeite Alternativen (Konnektionsmengen, Substitutionen, ...) simultan
 - Lasse alternative Beweiser parallel laufen
 - Die erste korrekte Lösung reicht als Beweis
- **Aufteilung in unabhängige Teilprobleme schwierig**
 - Unabhängige Probleme können isoliert gelöst werden
 - Abhängigkeiten zwischen Problemen benötigen Kommunikation (z.B. bei simultaner Unifikation mehrerer Konnektionen)

PARALLELITÄT: AUSNUTZUNG BESSERER HARDWARE

- **Leistungssteigerung durch Verdichtung hat Grenzen**
 - Es gibt kaum syntaktische Kriterien zur Auswahl der besten Strategie
 - Aussagenlogisches Beweisen ist bereits \mathcal{NP} -vollständig
 - Kann man Nichtdeterminismus ausnutzen?
- **Verteile alternative Wege auf mehrere Prozessoren**
 - Verarbeite Alternativen (Konnektionsmengen, Substitutionen, ...) simultan
 - Lasse alternative Beweiser parallel laufen
 - Die erste korrekte Lösung reicht als Beweis
- **Aufteilung in unabhängige Teilprobleme schwierig**
 - Unabhängige Probleme können isoliert gelöst werden
 - Abhängigkeiten zwischen Problemen benötigen Kommunikation (z.B. bei simultaner Unifikation mehrerer Konnektionen)
- **Praktische Effizienzsteigerungen selten erreicht**
 - Nur bei massiver Parallelität (mehrere tausend Prozessoren)

- **Erweiterung der Prädikatenlogik um ‘Modalitäten’**
 - Modellierung von Schlußfolgerungen, die im Alltag verwendet werden
 - Formel F ist beweisbar
 - Ich bin sicher oder glaube, daß F gilt
 - Möglicherweise ist F gültig

- **Erweiterung der Prädikatenlogik um ‘Modalitäten’**
 - Modellierung von Schlußfolgerungen, die im Alltag verwendet werden
 - Formel F ist beweisbar
 - Ich bin sicher oder glaube, daß F gilt
 - Möglicherweise ist F gültig
- **Syntax: Prädikatenlogik + Modaloperatoren \Box , \Diamond**
 - \Box , \Diamond sind Meta-Operatoren, die Aussagen über Formeln treffen
 - Lesart: $\Box F$: “notwendigerweise F ” $\Diamond F$: “möglicherweise F ”

- **Erweiterung der Prädikatenlogik um ‘Modalitäten’**
 - Modellierung von Schlußfolgerungen, die im Alltag verwendet werden
 - Formel F ist beweisbar
 - Ich bin sicher oder glaube, daß F gilt
 - Möglicherweise ist F gültig
- **Syntax: Prädikatenlogik + Modaloperatoren \Box , \Diamond**
 - \Box , \Diamond sind Meta-Operatoren, die Aussagen über Formeln treffen
 - Lesart: $\Box F$: “notwendigerweise F ” $\Diamond F$: “möglicherweise F ”
- **Semantik abhängig von vorgesehener Anwendung**
 - Je nachdem, ob \Box als “beweisbar”, “wissen”, “glauben” verstanden wird
 - $(\forall x \Box Px) \Rightarrow \Box(\exists x Px)$ ist nicht für jede Interpretation gültig

- **Erweiterung der Prädikatenlogik um ‘Modalitäten’**
 - Modellierung von Schlußfolgerungen, die im Alltag verwendet werden
 - Formel F ist beweisbar
 - Ich bin sicher oder glaube, daß F gilt
 - Möglicherweise ist F gültig
- **Syntax: Prädikatenlogik + Modaloperatoren \Box , \Diamond**
 - \Box , \Diamond sind Meta-Operatoren, die Aussagen über Formeln treffen
 - Lesart: $\Box F$: “notwendigerweise F ” $\Diamond F$: “möglicherweise F ”
- **Semantik abhängig von vorgesehener Anwendung**
 - Je nachdem, ob \Box als “beweisbar”, “wissen”, “glauben” verstanden wird
 - $(\forall x \Box Px) \Rightarrow \Box(\exists x Px)$ ist nicht für jede Interpretation gültig
- **Beweisverfahren: analog zur Konstruktiven Logik**
 - Modifizierter Konnektionsbeweiser mit Präfixen für Modaloperatoren
 - Präfix-Unifikation und Zulässigkeitstests für verschiedene Modallogiken

LOGIK HÖHERER STUFE (Einheit 17)

- **Konstrukte höherer Stufe kommen überall vor**

- Funktionen (2. Stufe): “Bestimme $x+2$ bei Eingabe x ”

$\lambda x.x+2$

LOGIK HÖHERER STUFE (Einheit 17)

- **Konstrukte höherer Stufe kommen überall vor**

- Funktionen (2. Stufe): “Bestimme $x+2$ bei Eingabe x ” $\lambda x.x+2$
- Induktionsprinzip: $\forall P.P(0) \wedge (\forall y:\mathbb{N}.P(y) \Rightarrow P(y+1)) \Rightarrow \forall x:\mathbb{N}.P(x)$

● Konstrukte höherer Stufe kommen überall vor

- Funktionen (2. Stufe): “Bestimme $x+2$ bei Eingabe x ” $\lambda x.x+2$
- Induktionsprinzip: $\forall P.P(0) \wedge (\forall y:\mathbb{N}.P(y) \Rightarrow P(y+1)) \Rightarrow \forall x:\mathbb{N}.P(x)$
- Zwischenwertsatz: $\forall f:\mathbb{R} \rightarrow \mathbb{R}.\forall a < b:\mathbb{R}.(f(a) > 0 \wedge f(b) < 0) \Rightarrow \exists x:\mathbb{R}.f(x)=0$

● Konstrukte höherer Stufe kommen überall vor

- Funktionen (2. Stufe): “Bestimme $x+2$ bei Eingabe x ” $\lambda x.x+2$
- Induktionsprinzip: $\forall P.P(0) \wedge (\forall y:\mathbb{N}.P(y) \Rightarrow P(y+1)) \Rightarrow \forall x:\mathbb{N}.P(x)$
- Zwischenwertsatz: $\forall f:\mathbb{R} \rightarrow \mathbb{R}.\forall a < b:\mathbb{R}.(f(a) > 0 \wedge f(b) < 0) \Rightarrow \exists x:\mathbb{R}.f(x)=0$
- Funktionale (3. Stufe): Ableitungsoperator $\lambda f.df/dx$

LOGIK HÖHERER STUFE (Einheit 17)

● Konstrukte höherer Stufe kommen überall vor

- Funktionen (2. Stufe): “Bestimme $x+2$ bei Eingabe x ” $\lambda x.x+2$
- Induktionsprinzip: $\forall P.P(0) \wedge (\forall y:\mathbb{N}.P(y) \Rightarrow P(y+1)) \Rightarrow \forall x:\mathbb{N}.P(x)$
- Zwischenwertsatz: $\forall f:\mathbb{R} \rightarrow \mathbb{R}.\forall a < b:\mathbb{R}.(f(a) > 0 \wedge f(b) < 0) \Rightarrow \exists x:\mathbb{R}.f(x)=0$
- Funktionale (3. Stufe): Ableitungsoperator $\lambda f.df/dx$

● Logik höherer Stufe hat keine Einschränkungen

- Extrem einfache Syntax: $x, \lambda x.t, f(a), \forall x P, P \Rightarrow Q$

● Konstrukte höherer Stufe kommen überall vor

- Funktionen (2. Stufe): “Bestimme $x+2$ bei Eingabe x ” $\lambda x.x+2$
- Induktionsprinzip: $\forall P.P(0) \wedge (\forall y:\mathbb{N}.P(y) \Rightarrow P(y+1)) \Rightarrow \forall x:\mathbb{N}.P(x)$
- Zwischenwertsatz: $\forall f:\mathbb{R} \rightarrow \mathbb{R}.\forall a < b:\mathbb{R}.(f(a) > 0 \wedge f(b) < 0) \Rightarrow \exists x:\mathbb{R}.f(x)=0$
- Funktionale (3. Stufe): Ableitungsoperator $\lambda f.df/dx$

● Logik höherer Stufe hat keine Einschränkungen

- Extrem einfache Syntax: $x, \lambda x.t, f(a), \forall x P, P \Rightarrow Q$
- Fester Auswertungsmechanismus (β -)Reduktion: $(\lambda x.t)(a) \longrightarrow t[a/x]$

● Konstrukte höherer Stufe kommen überall vor

- Funktionen (2. Stufe): “Bestimme $x+2$ bei Eingabe x ” $\lambda x.x+2$
- Induktionsprinzip: $\forall P.P(0) \wedge (\forall y:\mathbb{N}.P(y) \Rightarrow P(y+1)) \Rightarrow \forall x:\mathbb{N}.P(x)$
- Zwischenwertsatz: $\forall f:\mathbb{R}\rightarrow\mathbb{R}.\forall a < b:\mathbb{R}.(f(a)>0 \wedge f(b)<0) \Rightarrow \exists x:\mathbb{R}.f(x)=0$
- Funktionale (3. Stufe): Ableitungsoperator $\lambda f.df/dx$

● Logik höherer Stufe hat keine Einschränkungen

- Extrem einfache Syntax: $x, \lambda x.t, f(a), \forall x P, P \Rightarrow Q$
- Fester Auswertungsmechanismus (β -)Reduktion: $(\lambda x.t)(a) \longrightarrow t[a/x]$

● Logik höherer Stufe ist minimale Grundlagentheorie

- Keine Abstützung auf Mengentheorie erforderlich
- Reduktion erklärt Wert von Ausdrücken

● Konstrukte höherer Stufe kommen überall vor

- Funktionen (2. Stufe): “Bestimme $x+2$ bei Eingabe x ” $\lambda x.x+2$
- Induktionsprinzip: $\forall P.P(0) \wedge (\forall y:\mathbb{N}.P(y) \Rightarrow P(y+1)) \Rightarrow \forall x:\mathbb{N}.P(x)$
- Zwischenwertsatz: $\forall f:\mathbb{R} \rightarrow \mathbb{R}.\forall a < b:\mathbb{R}.(f(a) > 0 \wedge f(b) < 0) \Rightarrow \exists x:\mathbb{R}.f(x)=0$
- Funktionale (3. Stufe): Ableitungsoperator $\lambda f.df/dx$

● Logik höherer Stufe hat keine Einschränkungen

- Extrem einfache Syntax: $x, \lambda x.t, f(a), \forall x P, P \Rightarrow Q$
- Fester Auswertungsmechanismus (β -)Reduktion: $(\lambda x.t)(a) \longrightarrow t[a/x]$

● Logik höherer Stufe ist minimale Grundlagentheorie

- Keine Abstützung auf Mengentheorie erforderlich
- Reduktion erklärt Wert von Ausdrücken
- Mathematische Konzepte werden nicht über Axiome erklärt sondern als definatorische Abkürzung für komplexe Terme (logizistischer Ansatz)

● Konstrukte höherer Stufe kommen überall vor

- Funktionen (2. Stufe): “Bestimme $x+2$ bei Eingabe x ” $\lambda x.x+2$
- Induktionsprinzip: $\forall P.P(0) \wedge (\forall y:\mathbb{N}.P(y) \Rightarrow P(y+1)) \Rightarrow \forall x:\mathbb{N}.P(x)$
- Zwischenwertsatz: $\forall f:\mathbb{R}\rightarrow\mathbb{R}.\forall a < b:\mathbb{R}.(f(a)>0 \wedge f(b)<0) \Rightarrow \exists x:\mathbb{R}.f(x)=0$
- Funktionale (3. Stufe): Ableitungsoperator $\lambda f.df/dx$

● Logik höherer Stufe hat keine Einschränkungen

- Extrem einfache Syntax: $x, \lambda x.t, f(a), \forall x P, P \Rightarrow Q$
- Fester Auswertungsmechanismus (β -)Reduktion: $(\lambda x.t)(a) \longrightarrow t[a/x]$

● Logik höherer Stufe ist minimale Grundlagentheorie

- Keine Abstützung auf Mengentheorie erforderlich
- Reduktion erklärt Wert von Ausdrücken
- Mathematische Konzepte werden nicht über Axiome erklärt sondern als definatorische Abkürzung für komplexe Terme (logizistischer Ansatz)

● Extensionsverfahren ähnlich wie bei Prädikatenlogik

- Komplementaritätstest mit Unifikation höherer Stufe (unentscheidbar)

- **Andersartige Grundoperatoren**

- Additive, multiplikative und exponentielle Operatoren
- kommutative und nichtkommutative Versionen logischer Operatoren
- Idempotente und nicht-idempotente Versionen ($A \not\vdash A \wedge A$)
- Logik höherer Stufe
- Klassische und nichtklassische Logik simulierbar

- **Andersartige Grundoperatoren**
 - Additive, multiplikative und exponentielle Operatoren
 - kommutative und nichtkommutative Versionen logischer Operatoren
 - Idempotente und nicht-idempotente Versionen ($A \not\vdash A \wedge A$)
 - Logik höherer Stufe
 - Klassische und nichtklassische Logik simulierbar
- **Präzisere Verwaltung von Ressourcen (Linearität)**
 - Sequenzenkalkül ohne Kontraktion und Ausdünnung
 - Annahmen werden “verbraucht”
 - sehr kompliziert und von der Fachwelt nur wenig verstanden

- **Andersartige Grundoperatoren**
 - Additive, multiplikative und exponentielle Operatoren
 - kommutative und nichtkommutative Versionen logischer Operatoren
 - Idempotente und nicht-idempotente Versionen ($A \not\vdash A \wedge A$)
 - Logik höherer Stufe
 - Klassische und nichtklassische Logik simulierbar
- **Präzisere Verwaltung von Ressourcen (Linearität)**
 - Sequenzkalkül ohne Kontraktion und Ausdünnung
 - Annahmen werden “verbraucht”
 - sehr kompliziert und von der Fachwelt nur wenig verstanden
- **Extensionsverfahren muß linear sein**
 - Jedes Literal ist nur einmal konnektierbar
 - Automatisierung bisher nur für MLL / ALL gelungen
 - Matrixkalkül für MELL (ohne additive Operatoren) sehr aufwendig

HÖHERE FORMEN DES SCHLIESSENS

- **Analogieschließen**

- Übertragung einer Beweisidee auf neue, ähnliche Probleme
- Wichtig für Entdeckung neuer Beweise
- Konzept der “Ähnlichkeit” bisher schwer zu formulieren

HÖHERE FORMEN DES SCHLIESSENS

- **Analogieschließen**

- Übertragung einer Beweisidee auf neue, ähnliche Probleme
- Wichtig für Entdeckung neuer Beweise
- Konzept der “Ähnlichkeit” bisher schwer zu formulieren

- **Abstraktion**

- Anhebung der Ebene des Schließens von Literalen auf “Konzepte”
- Elimination überflüssiger formaler Details bei Beweissuche
- Wichtig für Verständnis und Planung von Beweisen
- Allgemeine Mechanismen bisher kaum verstanden

HÖHERE FORMEN DES SCHLIESSENS

● Analogieschließen

- Übertragung einer Beweisidee auf neue, ähnliche Probleme
- Wichtig für Entdeckung neuer Beweise
- Konzept der “Ähnlichkeit” bisher schwer zu formulieren

● Abstraktion

- Anhebung der Ebene des Schließens von Literalen auf “Konzepte”
- Elimination überflüssiger formaler Details bei Beweissuche
- Wichtig für Verständnis und Planung von Beweisen
- Allgemeine Mechanismen bisher kaum verstanden

● Beweisplanung

- Schematisierte Form der Abstraktion
- Makro-Operatoren formulieren abstraktere Beweisschritte
- KI-Planer sucht Kette von Operatoren, die zum Ziel führen können
- Assoziierte Beweistaktiken prüfen, ob Beweis im Detail funktioniert
- Teilerfolge im Zusammenhang mit unverdichteten Beweisen

HÖHERE FORMEN: VERWENDUNG DER META-EBENE

- **Taktiken: Meta-Programmierung von Strategien**
 - Formuliere Anweisungen zur Manipulation der Objektsprache in Meta-Sprache
 - **Sicher**: Meta-Programme müssen im Endeffekt Basisinferenzen verwenden
 - **Flexibel**: Benutzer können eigene deduktive Strategien ergänzen

HÖHERE FORMEN: VERWENDUNG DER META-EBENE

- **Taktiken: Meta-Programmierung von Strategien**
 - Formuliere Anweisungen zur Manipulation der Objektsprache in Meta-Sprache
 - **Sicher:** Meta-Programme müssen im Endeffekt Basisinferenzen verwenden
 - **Flexibel:** Benutzer können eigene deduktive Strategien ergänzen
- **Meta-Inferenz: Beweise Aussagen über Strategien**
 - Formuliere Aussagen über Zusammenhang zwischen Logik und Strategie
 - Welche Schlüsse sind in der Logik überhaupt erlaubt?
 - Für welche (syntaktischen) Formelklassen sind die Strategien erfolgreich
 - Formulierung in separater Meta-Logik (Logical Framework)

HÖHERE FORMEN: VERWENDUNG DER META-EBENE

- **Taktiken: Meta-Programmierung von Strategien**
 - Formuliere Anweisungen zur Manipulation der Objektsprache in Meta-Sprache
 - **Sicher:** Meta-Programme müssen im Endeffekt Basisinferenzen verwenden
 - **Flexibel:** Benutzer können eigene deduktive Strategien ergänzen
- **Meta-Inferenz: Beweise Aussagen über Strategien**
 - Formuliere Aussagen über Zusammenhang zwischen Logik und Strategie
 - Welche Schlüsse sind in der Logik überhaupt erlaubt?
 - Für welche (syntaktischen) Formelklassen sind die Strategien erfolgreich
 - Formulierung in separater Meta-Logik (Logical Framework)
- **Reflektion: Meta-Inferenz in der Objektlogik**
 - Möglich, wenn Meta- und Objektsprache syntaktisch ähnlich
 - Erlaubt Beweise über Strategien innerhalb der Objektlogik
 - Nur eingeschränkt in Logik erster Stufe möglich
 - Mechanismen bisher zu wenig verstanden

HÖHERE FORMEN: SEMANTISCHE SUCHFÜHRUNG

- **Menschliches Schließen basiert auf Bedeutung**
 - Syntaktische Struktur wird nur als Beschreibungsform verstanden
 - Veranschaulichung (Betrachtung einfacher Modelle) liefert Beweisidee
 - Konkreter Beweis prüft, ob Beweisidee im Detail tragfähig ist oder verfeinert, modifiziert, verworfen werden muß

HÖHERE FORMEN: SEMANTISCHE SUCHFÜHRUNG

- **Menschliches Schließen basiert auf Bedeutung**
 - Syntaktische Struktur wird nur als Beschreibungsform verstanden
 - Veranschaulichung (Betrachtung einfacher Modelle) liefert Beweisidee
 - Konkreter Beweis prüft, ob Beweisidee im Detail tragfähig ist oder verfeinert, modifiziert, verworfen werden muß
- **Unterstütze Beweissuche durch semantische Analyse**
 - Untersuche konkrete Modelle (Beispiele) einer Formel
 - Eliminiere Teilziele, die von diesen Beispielen widerlegt werden
 - Beende zugehörigen Pfad der Beweissuche als erfolglos
 - Setze zurück und beginne alternativen Suchpfad

HÖHERE FORMEN: SEMANTISCHE SUCHFÜHRUNG

- **Menschliches Schließen basiert auf Bedeutung**
 - Syntaktische Struktur wird nur als Beschreibungsform verstanden
 - Veranschaulichung (Betrachtung einfacher Modelle) liefert Beweisidee
 - Konkreter Beweis prüft, ob Beweisidee im Detail tragfähig ist oder verfeinert, modifiziert, verworfen werden muß
- **Unterstütze Beweissuche durch semantische Analyse**
 - Untersuche konkrete Modelle (Beispiele) einer Formel
 - Eliminiere Teilziele, die von diesen Beispielen widerlegt werden
 - Beende zugehörigen Pfad der Beweissuche als erfolglos
 - Setze zurück und beginne alternativen Suchpfad
- **Erfolgreich für Geometrische Beweisverfahren**
 - Basismethode von Gelernter (1959) formuliert für Horn Formeln
 - Erweiterbar auf volle Prädikatenlogik
 - Automatisierung schwierig, da Erzeugung von Gegenbeispielen erforderlich

ANWENDUNGEN

- **Logische Programmiersprachen (Prolog)**
 - Hornklausellogik + Kontrollmechanismen (cut!)
 - Turingmächtig, effiziente “Compiler” vorhanden

ANWENDUNGEN

- **Logische Programmiersprachen (Prolog)**
 - Hornklausellogik + Kontrollmechanismen (cut!)
 - Turingmächtig, effiziente “Compiler” vorhanden
- **Formale Mathematik**
 - Automatische Beweise für mathematische Teiltheorien
 - Interaktive Beweise für jegliche Form von Mathematik

ANWENDUNGEN

- **Logische Programmiersprachen (Prolog)**
 - Hornklausellogik + Kontrollmechanismen (cut!)
 - Turingmächtig, effiziente “Compiler” vorhanden
- **Formale Mathematik**
 - Automatische Beweise für mathematische Teiltheorien
 - Interaktive Beweise für jegliche Form von Mathematik
- **Programmverifikation**
 - Höchste Stufe des Software-TÜV
 - Prüfe Korrektheit, Zuverlässigkeit, Sicherheit, ...

ANWENDUNGEN

- **Logische Programmiersprachen (Prolog)**
 - Hornklausellogik + Kontrollmechanismen (cut!)
 - Turingmächtig, effiziente “Compiler” vorhanden
- **Formale Mathematik**
 - Automatische Beweise für mathematische Teiltheorien
 - Interaktive Beweise für jegliche Form von Mathematik
- **Programmverifikation**
 - Höchste Stufe des Software-TÜV
 - Prüfe Korrektheit, Zuverlässigkeit, Sicherheit, ...
- **Programmsynthese**
 - Erzeugung korrekter Algorithmen aus formalen Spezifikationen
 - Programmverifikation während der Programmentwicklung

ANWENDUNGEN

- **Logische Programmiersprachen (Prolog)**
 - Hornklausellogik + Kontrollmechanismen (cut!)
 - Turingmächtig, effiziente “Compiler” vorhanden
- **Formale Mathematik**
 - Automatische Beweise für mathematische Teiltheorien
 - Interaktive Beweise für jegliche Form von Mathematik
- **Programmverifikation**
 - Höchste Stufe des Software-TÜV
 - Prüfe Korrektheit, Zuverlässigkeit, Sicherheit, ...
- **Programmsynthese**
 - Erzeugung korrekter Algorithmen aus formalen Spezifikationen
 - Programmverifikation während der Programmentwicklung
- **Wissensrepräsentation & Expertensysteme**

ANWENDUNGEN

- **Logische Programmiersprachen (Prolog)**
 - Hornklausellogik + Kontrollmechanismen (cut!)
 - Turingmächtig, effiziente “Compiler” vorhanden
- **Formale Mathematik**
 - Automatische Beweise für mathematische Teiltheorien
 - Interaktive Beweise für jegliche Form von Mathematik
- **Programmverifikation**
 - Höchste Stufe des Software-TÜV
 - Prüfe Korrektheit, Zuverlässigkeit, Sicherheit, ...
- **Programmsynthese**
 - Erzeugung korrekter Algorithmen aus formalen Spezifikationen
 - Programmverifikation während der Programmentwicklung
- **Wissensrepräsentation & Expertensysteme**

mehr als ein akademisches Spielzeug

VERTIEFUNGSMÖGLICHKEITEN UND ERGÄNZUNGEN

- **Seminar: Automatisches Beweisen in der Aussagenlogik**
 - Hocheffiziente Verfahren für aussagenlogisches Beweisen

VERTIEFUNGSMÖGLICHKEITEN UND ERGÄNZUNGEN

- **Seminar: Automatisches Beweisen in der Aussagenlogik**
 - Hocheffiziente Verfahren für aussagenlogisches Beweisen
- **Automatisierte Logik und Programmierung** (WS07/08)
 - Konstruktive Typentheorie (Logik höherer Stufe + ...)
 - Beweiseditoren, Taktiken, Entscheidungsprozeduren
 - Automatisierte Softwareentwicklung

VERTIEFUNGSMÖGLICHKEITEN UND ERGÄNZUNGEN

- **Seminar: Automatisches Beweisen in der Aussagenlogik**
 - Hocheffiziente Verfahren für aussagenlogisches Beweisen
- **Automatisierte Logik und Programmierung** (WS07/08)
 - Konstruktive Typentheorie (Logik höherer Stufe + ...)
 - Beweiseditoren, Taktiken, Entscheidungsprozeduren
 - Automatisierte Softwareentwicklung
- **Vorlesungen von Dr. E. Richter und Prof. T. Schaub**
 - Fuzzy Logik, Nicht-monotones Schließen,
 - Logikprogrammierung, KI, Wissensrepräsentation, ...

VERTIEFUNGSMÖGLICHKEITEN UND ERGÄNZUNGEN

- **Seminar: Automatisches Beweisen in der Aussagenlogik**
 - Hocheffiziente Verfahren für aussagenlogisches Beweisen
- **Automatisierte Logik und Programmierung** (WS07/08)
 - Konstruktive Typentheorie (Logik höherer Stufe + ...)
 - Beweiseditoren, Taktiken, Entscheidungsprozeduren
 - Automatisierte Softwareentwicklung
- **Vorlesungen von Dr. E. Richter und Prof. T. Schaub**
 - Fuzzy Logik, Nicht-monotones Schließen,
 - Logikprogrammierung, KI, Wissensrepräsentation, ...
- **Mitarbeit in der Forschung** (Studien-/Diplomarbeiten)
 - Problemsammlungen, Benchmarks von Beweisern \mapsto J. Otten, T. Rath
 - Implementierung effizienter Matrixbeweiser \mapsto J. Otten, T. Rath
 - Theorieschließen, Entscheidungsprozeduren, Rewrite-Verfahren \mapsto C. Kreitz
 - Anwendungen: Synthese, Verifikation, Sicherheit, ... \mapsto C. Kreitz, K. Peters
 - \vdots