Praktische Informatik 2
Sommer-Semester 2005

Prof. Dr. sc. Hans-Dieter Burkhard

www.ki.informatik.hu-berlin.de

Repräsentation von „Wissen“

- Wissen kann in unterschiedlichen Formen dargestellt (repräsentiert) werden.
- Die Syntax bestimmt die formale Struktur.
- Erst die Semantik bestimmt die Bedeutung.

- Fragen zum Nachdenken: Unter welchen Gesichtspunkten wird die Struktur (Syntax) festgelegt? Welche Rolle spielt dabei die Semantik? Welche Anforderungen ergeben sich für die maschinelle Verarbeitung?
- *.....(Notieren Sie sich eigene Fragen!)*
- .....

Explizites vs. Implizites Wissen

- Aus vorhandenem (explizitem) Wissen kann auf neues (implizites) Wissen geschlossen werden.
- Inferenz bezeichnet den Prozess des Herleitens von implizitem Wissen.

- Bemerkung: Im Wissensmanagement bezeichnet „explizites Wissen“ das dokumentierte Wissen, „implizites Wissen“ dagegen nicht verbalisierte Fähigkeiten.
- *.....(Notieren Sie sich eigene Bemerkungen!)*
- .....

Fakten in Prolog

- Ein Fakt (Datensatz) hat die Form funktor(Argumente) .
- funktor ist der Name einer n-stelligen Relation (Prädikat).
- Ein Funktor ist bestimmt durch Namen und Stelligkeit.

- Endliche Relationen können durch Aufzählung der Fakten in Prolog gespeichert werden (Datenbank).

Anfragen

- Anfragen haben die Form ?-funktor(Argumente) .
- Argumente können dabei auch Variable (großer Anfangsbuchstabe) sein.
- Wenn ein passender Fakt in der Datenbasis enthalten ist, gibt Prolog eine positive Antwort.
- Wenn die Anfrage Variablen enthält, werden passende Werte eingesetzt.

- Frage zum Nachdenken: Was ist ein „passender Fakt“?

CWA: Closed World Assumption

- Wann gibt Prolog keine positive Antwort?
- Annahme einer abgeschlossenen Welt: *Nur was ich weiß, ist wahr.*

- Fragen zum Nachdenken: Was ist keine „positive Antwort“? Was weiß Prolog?

Relationen definieren durch Regeln

- Man kann aus gegebenen Relationen (Prädikaten) neue Relationen definieren mithilfe von Regeln.
- Regeln haben die Form Kopf :- Körper.
- Intuitive Semantik: Der Aussage im Kopf gilt, wenn alle Aussagen im Körper gelten.

- Fragen zum Nachdenken: Wie ist die Syntax (Struktur) für Regeln? Ist Ihre Beschreibung so beschaffen, dass auch ein Rechner sie nachprüfen kann?

Regel als logische Formel

- Eine Regel entspricht einer Implikation Körper -> Kopf.
- Die Variablen in Regeln sind universell quantifiziert.
- Die mit Komma getrennten Relationen der rechten Seite sind konjunktiv verknüpft.
- Alternativen werden durch mehrere Regeln mit dem gleichen Kopf-Funktor ausgedrückt.

- Fragen zum Nachdenken: Eine Relation ist eine Menge von Fakten. Was bedeuten die Regeln mengentheoretisch?

Regel als logische Formel

- Relationen (Prädikate) sind atomare Formeln.
- Positives Literal: atomare Formel.
- Negatives Literal: negierte atomare Formel.
- Hornklausel: Alternative mit maximal einem positiv. Literal.
- Regeln sind Hornklauseln.

Regel als logische Formel

- Intuitive Bedeutung: Regel hat Form goal :- subgoals.
- Dabei soll goal bewiesen werden, indem alle subgoals bewiesen werden (Analogie: Abtrennungsregel).
- Prolog kann entsprechende Beweise (Ableitungen) durchführen.

Ableitung in Prolog

- Prolog versucht, sukzessive geeignete Klauseln für die Erfüllung der subgoals zu finden.
- Wenn das gelingt, hat Prolog einen Beweisbaum konstruiert. Der Beweisbaum enthält die benutzten Aussagen (geeignete Relationen mit geeigneten Argumenten) in der Reihenfolge ihrer Verwendung.

- Fragen zum Nachdenken: Wie kann das programmtechnisch umgesetzt werden?

Prolog-Programm

- Programme bestehen aus Klauseln.
- Klauseln sind Fakten oder Regeln.
- Prozedur: Menge von Klauseln mit gleichem Kopf-Funktor.
- Klauseln/Prozeduren definieren Relationen (Prädikate).

Prolog-Programm, Interpreter

- Anfragen an Programme haben die Form ?-goal(Argumente). mit der Bedeutung: „Gibt es Werte, so dass die Anfrage erfüllt wird?“
- Wenn das Prolog-System einen Beweis findet, erfolgt eine positive Antwort mit entsprechenden Werten.
- Interpreter: Laufzeit-System für Prolog, das Antworten auf Anfragen an ein Programm gibt, d.h. Beweise sucht und ausführt (Theorem-Beweiser).

- Fragen zum Nachdenken (Wiederholung): Wie kann das programmtechnisch umgesetzt werden? Welche unterschiedlichen Möglichkeiten gibt es? Geht das überhaupt?

Prolog und Logik

- Prolog-Programm P = Axiome
- Anfrage Q = Frage nach Beweisbarkeit von Q aus P
- Antwort = Ergebnis eines Beweises bzw. Fehlschlag
- Trace = Verlauf des Beweisversuchs

Prolog-Interpreter und PK1

- Ziel: Prolog-Interpreter als universelles Beweis-Verfahren im PK1. Erwünschte Eigenschaften:
  - **Vollständigkeit**:
    Interpreter liefert alle Folgerungen aus dem Programm.
  - **Korrektheit**:
    Interpreter liefert nur Folgerungen aus dem Programm.
- Allgemeingültigkeit eines Ausdrucks H im PK1 ist nur partiell entscheidbar (aufzählbar), aber nicht entscheidbar.

- Fragen zum Nachdenken: Was bedeutet das für Prolog?

Interpreter für Prolog

- Idee:
  (Systematisches) Probieren aller Beweismöglichkeiten.
- Der Interpreter verwaltet Listen von offenen Teil-Zielen.
- Wenn Liste leer ist, ist der Beweis fertig.
- Veränderungen der Liste erfolgen durch Anwendung von passenden Klauseln.

- Fragen zum Nachdenken: Wann schlägt ein Beweisversuch fehl?

Interpreter für Standard-Prolog

- Systematisches Probieren von Beweismöglichkeiten nach folgendem Schema:
  - Reihenfolge innerhalb einer Prozedur: oben vor unten.
  - Reihenfolge innerhalb einer Klausel: links vor rechts.
- Organisation mithilfe von Backtracking (Verwendung einer alternativen Klausel).

- Fragen zum Nachdenken: Hat das Auswirkungen auf Korrektheit und Vollständigkeit? Was für ein Suchverfahren wird damit implementiert?

Unifikation (matching, instantiation)

- Präzisierung für „passende“ Klausel.
- Zwei Terme (Strukturen) sind unifizierbar, wenn sie mit Hilfe eines Unifikators (Variablensubstitution) als Zeichenkette identisch gemacht werden können.
- Der resultierende Term ist die „Instantiierung“ (ein gemeinsames Beispiel für beide Terme).

- Fragen zum Nachdenken: Kann es unterschiedliche Instantiierungen geben? Wie werden die Begriffe „Term“ und „Struktur“ in Prolog gebraucht? Gibt es Unterschiede zum Gebrauch im PK1?

Unifikation (matching, instantiation)

- Beweisen eines Teilziels erfordert „Matchen“ von Teilziel und Klauselkopf.
- Analogie: „Prozedur-Aufruf“, Parameterübergabe durch Instanziierung.
- Bindung statt Wertzuweisung.

- Fragen zum Nachdenken: Variable „gehören“ den Klauseln, was bedeutet das für Lebensdauer und Sichtbarkeit? Müssen die Variablen beim Aufruf der Prozedur an Werte gebunden sein?

Prolog-Operatoren „=“ und „==“

- Prolog enthält auch Operatoren für den Vergleich von Strukturen, mit denen entsprechende Teilziele aufgerufen werden können.
- `t1 = t2` ist erfüllt, falls `t1` und `t2` unifizierbar sind.
- `t1 == t2` ist erfüllt, falls `t1` und `t2` identisch sind.

Fragen zum Nachdenken: Worin unterscheidet sich die Anwendung dieser Operatoren von einem Klauselaufruf? Warum schlägt das Ziel „`7 = 3+4`“ fehl?

Occur-Check

- Variablen-Substitutionen können auch rekursiv sein: σ(X) = funktor(..., X, ...).
- Das erfordert ggf. aufwendige Auswertungsverfahren.
- Unterschiedliche Reaktionen von Prolog-Systemen auf Anfragen.

Problemzerlegung

- Komplexe Probleme können oft durch Aufgliederung in Teilprobleme gelöst werden.
- Problemzerlegung kann als Modell für Prolog verstanden werden.
- Die Lösungen der Teilprobleme werden zu einer Gesamtlösung verbunden. Bei gleichen Variablen müssen die Bindungen übereinstimmen.

Fragen zum Nachdenken: Wann müssen die Werte für Variable bekannt sein?

Und-Oder-Baum

- Eine Problemzerlegung kann durch einen Und-Oder-Baum modelliert werden.
- Knoten können nach ihrer Stellung und Bedeutung klassifiziert werden.
- Eine weitere Klassifizierung ergibt sich hinsichtlich lösbarer bzw. unlösbarer Knoten.

Fragen zum Nachdenken: Treten Probleme auf bzgl. der Klassifizierung „lösbar/unlösbar“, wenn beliebige Grafen bzw. unendliche Bäume anstelle von endlichen Bäumen betrachtet werden?

Lösungsbaum

- Der Lösungsbaum entspricht dem Beweisbaum von Prolog.
- Bei endlichem Und-Oder-Baum können Lösungsbäume im Falle der Existenz konstruiert werden (z.B. bottom-up).
- Mithilfe einer Listenstruktur ist auch eine Vorwärtssuche mit Backtracking möglich.
- Der Interpreter für Standard-Prolog realisiert eine solche Suche mit speziellen Reihenfolge-Vorgaben.

Interpreter für Standardprolog

- Der Interpreter verwaltet zwei verschachtelte Keller während der Suche nach einem Beweis:
  - Liste der Prozedur-Aufrufe (Klauseln).
  - Liste der offenen Teilziele.
- Beim einem erfolgreichen Beweis ist die Liste der offenen Teilziele leer, die Liste der Prozeduraufrufe i.a. nicht leer.
- Beim Abschluss der Beweisversuche ist die Liste der Prozeduraufrufe leer, die Liste der offenen Teilziele nicht.
- Beide Listen können verschachtelt werden.

- Fragen zum Nachdenken: Wie hängen die Möglichkeiten für das Backtracking mit den Listen zusammen?

Interpreter für Standardprolog

- Für jeden Prozeduraufruf wird ein spezieller Speicherbereich („frame“) mit notwendigen Informationen für das Laufzeitsystem angelegt.
- Die Argumente der Prozedur werden im „environment“ (einem speziellen Bereich des frame) verwaltet.
- Die Bindungen der Argumente werden über Referenzen im environment verwaltet.

- Fragen zum Nachdenken: Wie kann verhindert werden, dass beim Backtracking „dangling references“ entsehen?