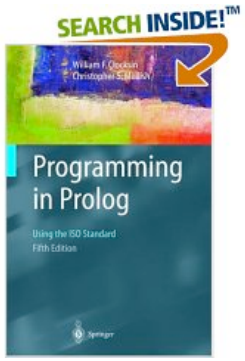
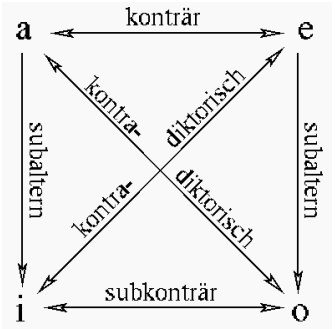
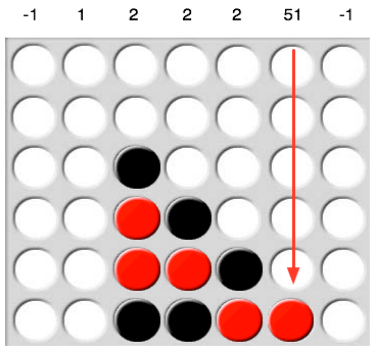






Prolog

Eignung logischer Programmiersprachen für Spiele-KI-Entwicklung am Beispiel Prolog

Vortrag Seminar Spiele-KI im SS2007 an der Fachhochschule Wedel
27.6.2007
Max Rohde, winf2848

Motivation

fhwedel
UNIVERSITY OF APPLIED SCIENCES

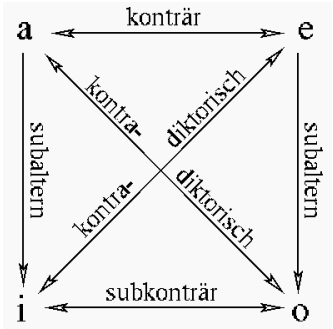
- Beschäftigung mit einem alternativen Konzept für Programmiersprachen (imparativ/objektorientiert, funktional, logisch)
- Bewertung dieses Konzeptes ...
- ... besonders unter Beachtung des praktischen Anwendungsgebietes Spiele-KI

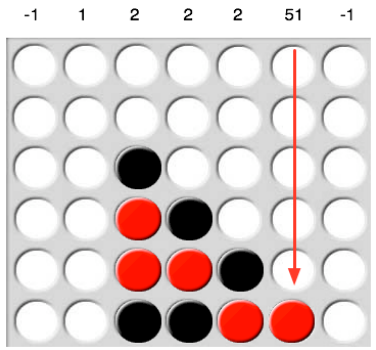
Ziel:

- Verständnis der grundlegenden Arbeitsweise von Prolog
- Um Bewertung der Eignung nachvollziehen zu können

Agenda

fhwedel
UNIVERSITY OF APPLIED SCIENCES





I. Grundlegende Eigenschaften von Prolog

II. Von Aussagen zur Prolog-Syntax

III. Wie Prolog arbeitet

IV. Vor- und Nachteile Prolog

V. Beispiel für Spiele-KI in Prolog 

Agenda

fhwedel
UNIVERSITY OF APPLIED SCIENCES

I. Grundlegende Eigenschaften von Prolog

1. Relational
2. Logisch
3. Deklarativ

II. Von Aussagen zur Prolog-Syntax

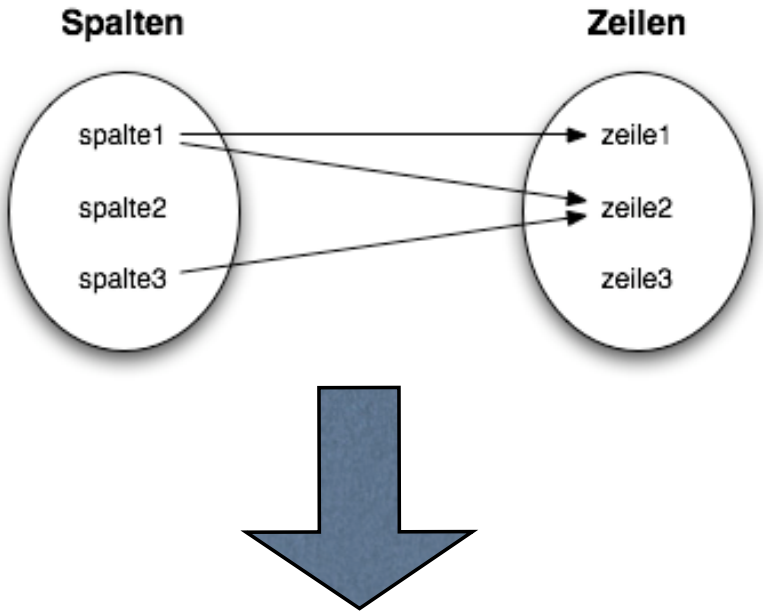
III. Wie Prolog arbeitet

IV. Vor- und Nachteile Prolog

V. Beispiel für Spiele-KI in Prolog

1. Relational

fhwedel
UNIVERSITY OF APPLIED SCIENCES



felder

Prolog:

```
feld(spalte1, zeile1).
feld(spalte1, zeile2).
feld(spalte3, zeile2).
```

SQL:

```
SELECT * FROM FELDER WHERE spalte='1'

spalte   zeile
------------
   1      1
   1      2
```

2. Logisch

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Eine logische Programmiersprache beschreibt im weitesten Sinne eine Programmiersprache die sich der mathematischen Logik bedient ...

Java Quellcode:

```
boolean regen=true;
boolean keinRegenschirm=false;
nass = regen && keinRegenschirm;
```

- ... im engeren Sinne jedoch im Gegensatz zu einer imparativen Programmiersprache keine **Folge** von Anweisungen enthält, sondern aus einer **Menge** von Axiomen und Schlussregeln besteht.

3. Deklarativ

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Nicht wie, sondern was!
- Beispiel: Arithmetik in imparativen Programmier-sprachen

Java Quellcode:

```
int i=5;
int j=3;
int k=i*j+j;
```

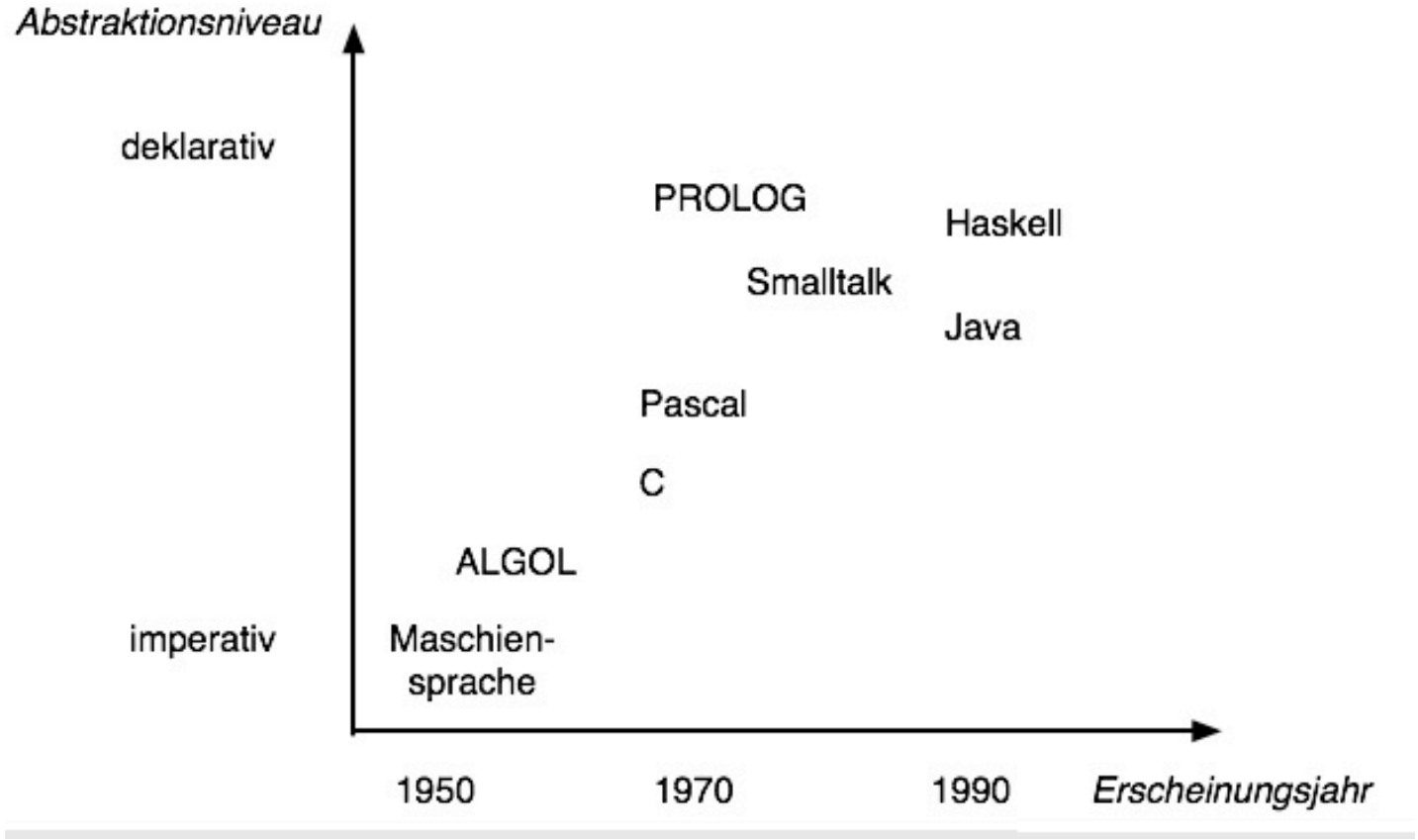
- Deklaratives Programm zum Sortieren:

```
Eine Liste „Sortiert“ ist genau dann eine sortierte Version einer Liste
„Ausgangsliste“, wenn „Ausgangsliste“ eine Permutation der Liste
„Sortiert“ ist und die Liste „Sortiert“ sortiert ist.
```

3. Deklarativ

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Entwicklung deklarativer Programmiersprachen



Eigene Graphik

Agenda

fhwedel
UNIVERSITY OF APPLIED SCIENCES

I. Grundlegende Eigenschaften von Prolog

II. Von Aussagen zur Prolog-Syntax

1. Atome
2. Fakten
3. Variablen
4. Regeln
5. Rekursion
6. Abfragen

4. Regeln

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Wenn X eine Spalte ist und Y eine Zeile, dann bilden Sie zusammen ein Feld.

Prädikatenlogik:

```
spalte(x) ∧ zeile(y) ⇒ feld(x, y)
```

∧ wird zu: ,

⇐ wird zu: :-

Prolog:

```
feld(X, Y) :- spalte(X), zeile(Y).
```

4. Regeln

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- An was erinnert uns das?
- Klauseln:

$\forall\ x_1\ \ldots\ \forall\ x_n : L_1 \vee \ldots \vee L_m$
mit $n \geq 1$, $m \geq 1$ und $x_{1..n}$ als einzige Variablen in $L_{1..n}$

- Horn-Klauseln

$[L_1 \wedge \ldots \wedge Lm] \Rightarrow S$

$\neg L_2 \vee \ldots \vee \neg L_m \vee L_1$
mit $n \geq 1$, $m \geq 2$ und $x_{1..n}$ als einzige Variablen in $L_{1..n}$

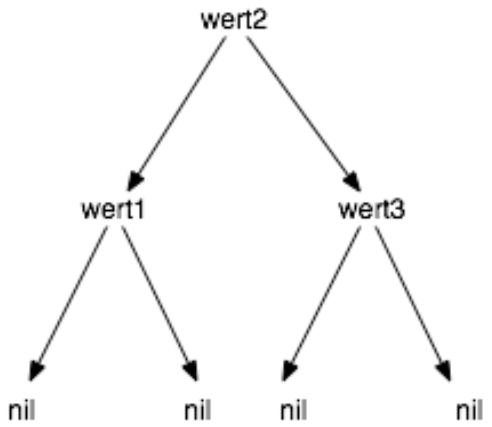
- Erfüllbarkeit in polynominaler Laufzeit nachweisbar!

5. Rekursion

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Rekursive Strukturen
- Beispiel: Binärer Baum

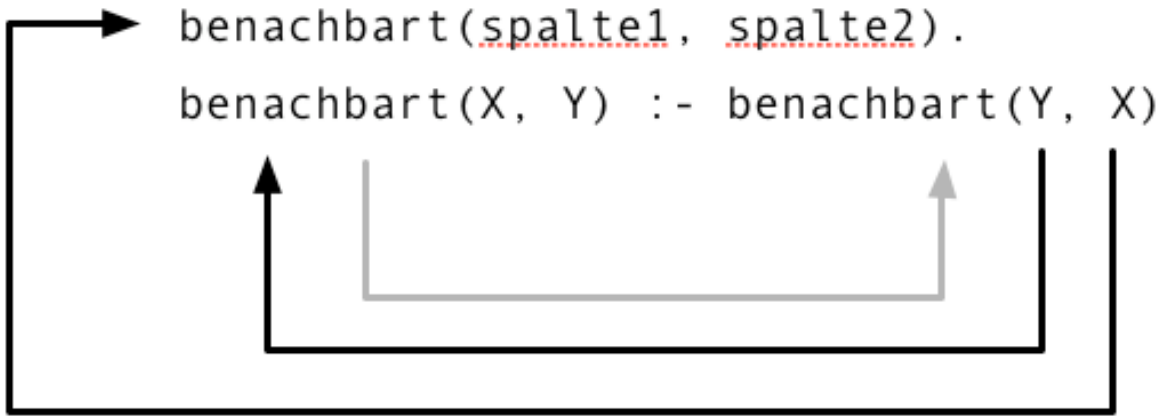
```
knoten(
   knoten(nil, wert1, nil),
   wert2,
   knoten(nil, wert3, nil)).
```



5. Rekursion

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Rekursive Auswertung



6. Abfragen

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Sind spalte1 und spalte2 benachbart?
- Welche Spalten sind mit spalte2 benachbart?

Agenda

fhwedel
UNIVERSITY OF APPLIED SCIENCES

I. Grundlegende Eigenschaften von Prolog

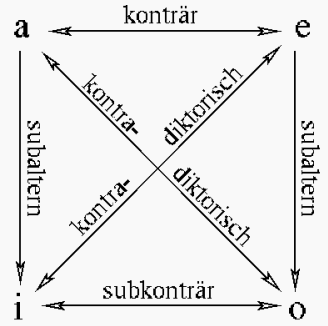
II. Von Aussagen zur Prolog-Syntax

III. Wie Prolog arbeitet

1. Unifikation
2. Backtracking

IV. Vor- und Nachteile Prolog

V. Beispiel für Spiele-KI in Prolog



1. Unifikation

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Substitution

$\sigma = \{ X_1 \rightarrow t_1, \ldots, X_n \rightarrow t_m \}$ mit Variablen X_n und Termen t_m, $n \geq 0$, $m \geq 0$

- Unifikation

Zwei Terme t_1 und t_2 sind unifizierbar, wenn eine Substitution σ existiert, die die beiden Terme identisch macht: $\sigma(t_1) = \sigma(t_2)$

- Ist die Wahl der σ eindeutig?
- Bildung des Most General Unifiers

Unifikation mit der Substitution σ, die die minimale Anzahl an Substitutions-Operationen $X \rightarrow t$ aufweist.

1. Unifikation

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Anfrage stellen

```
?- feld(spalte1, zeile1).
```

- Suche nach passenden Regeln/ Fakten

```
Finde diejenige Teilmenge T aus allen Klauseln des Programms für die gilt:
Es existiert ein σ, so dass gilt:
σ(feld(spalte1, zeile1)) = σ(Kopf(Tn))
```

- feld/2 gefunden

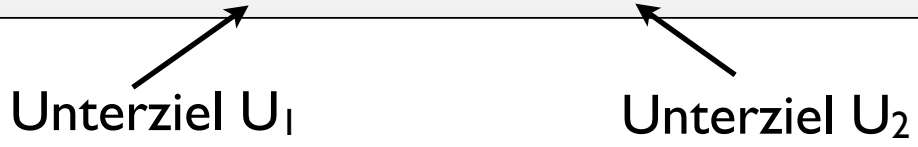
```
σ(feld(spalte1, zeile1)) = σ(Kopf(feld(X, Y) :- spalte(X), zeile(Y)))
mit σ = { X → spalte1, Y → zeile1 }
damit T = { feld(X, Y) :- spalte(X), zeile(Y) }
```

1. Unifikation

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Da eine Regel gefunden wurde, müssen die Unterziele überprüft werden!

```
feld(spalte1, zeile1) :- spalte(spalte1), zeile(zeile1).
```



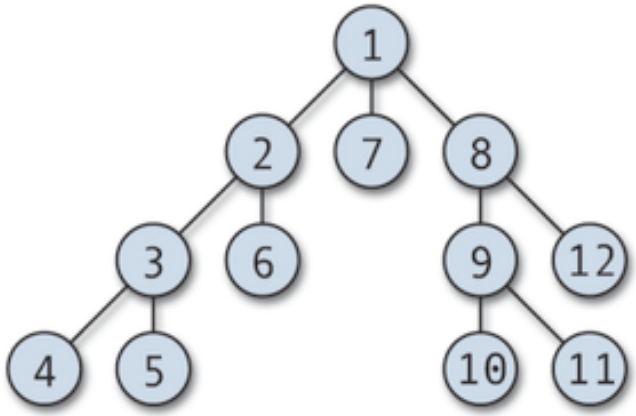
- Dabei wird analog für jedes Unterziel von links nach rechts vorgegangen.

- Doch was ist, wenn mehr als eine passende Klausel gefunden wurde? Also |T| > 1.

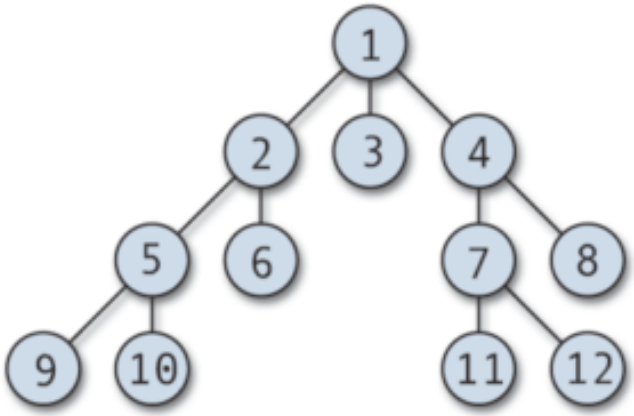
2. Backtracking

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Depth-First, Leftmost-First Auswertung des Lösungsbaumes



Tiefensuche



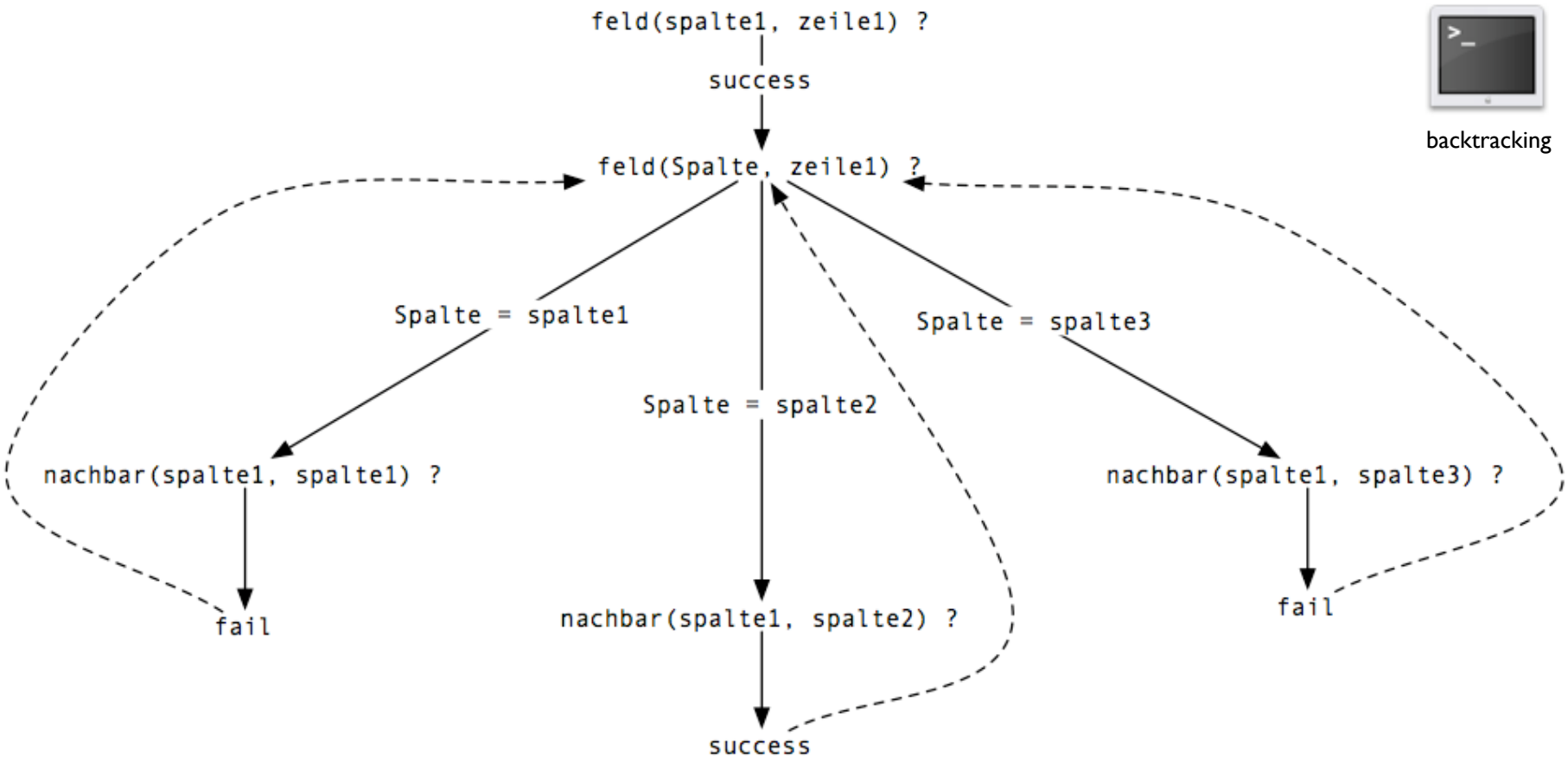
Breitensuche

- Alternativen entstehen dabei, wenn mehr als eine passende Klausel gefunden wurde.

2. Backtracking

fhwedel
UNIVERSITY OF APPLIED SCIENCES

```
?- links_von(feld(spalte1, zeile1), feld(Spalte, Zeile)).
```



backtracking

Agenda

fhwedel
UNIVERSITY OF APPLIED SCIENCES

III. Wie Prolog arbeitet

IV.Vor- und Nachteile Prolog

I. Allgemein

a. Allgemeine Vorteile

b. Allgemeine Nachteile

II. Für Spiele-KI

a. Vorteile für Spiele-KI

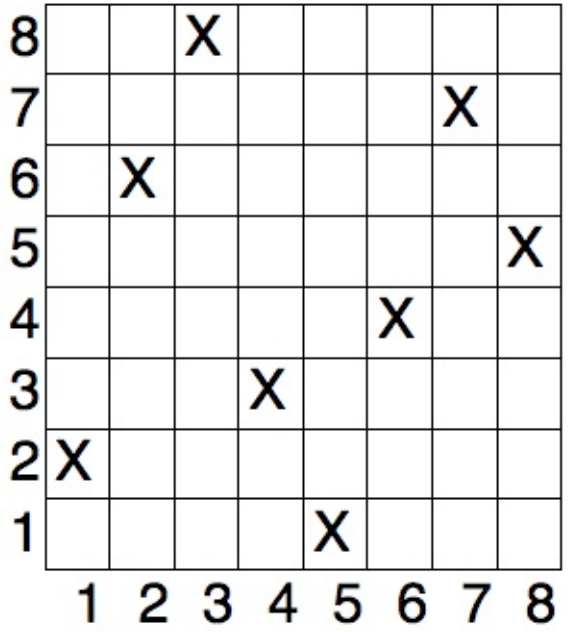
b. Nachteile für Spiele-KI

I. a. allgemeine Vorteile

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Geeignet bei brute force:

- Geeignet für Probleme, die sich mit Backtracking (bzw. brute force) lösen lassen
- z. B. 8-Damen-Problem



I. a. allgemeine Vorteile

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Allgemeinere Programme:

- Bei rein logischer Programmierung entstehen allgemeinere Programme
- Unsortierte Listen
- 8-Damen: Wie viele Damen wurden positioniert?

```
?- damen(N, [], [p(1,4),p(2,2),p(3,7),p(4,3),p(5,6),p(6,8),p(7,5),p(8,1)]).
```

I. b. allgemeine Nachteile

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Softwaretechnische Mängel:

- Globaler Namensraum für Prädikate
- Nur rudimentäres Modularisierungskonzept
- Ungetypt

I. b. allgemeine Nachteile

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Probleme mit Negation:

- Bedeutung des Prädikates not/1 entspricht nicht logischem nicht

```
?-not(nachbar(spalte1, X)), spalte(X).
Failure

?-not(nachbar(spalte1, spalte3)), spalte(spalte3).
Success
```

- Funktioniert nur bei instanzierten Varaiblen

I. b. allgemeine Nachteile

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Arithemtische Ausdrücke :

- Verwendung arithmetischer Operatoren führt zu nicht mehr logischen Programmen

```
?- 2+5 = 5+2
Failure

?-Links is 2+5, Rechts is 5+2, Links=Rechts
Success


plus_1(Zahl, ZahlPlus1) :- ZahlPlus1 is Zahl + 1.
```

- Bei plus_1 ist Zahl Inputparameter und ZahlPlus1 Outputparameter --> nicht mehr relational!

1. b. allgemeine Nachteile

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Keine deklarative Programmierung:

- Eine rein deklarative Programmierung ist in Prolog meist **nicht** möglich
- Gründe: Unvollständigkeit, Negation (Reiheinfolge) und Performance

Unvollständig:

```
benachbart(X, Y) :- benachbart(Y, X).


?- benachbart(spalte1, X)
... unendliche lange Auswertung
```

I. b. allgemeine Nachteile

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Keine deklarative Programmierung:

- Negation (Reiheinfolge)

```
?-not(nachbar(spalte1, X)), spalte(X).
Failure


?-spalte(X), not(nachbar(spalte1, X)).
X=spalte1, X=spalte3
```

- Das Gleiche gilt für den Operator \=/2 und arithmetische Operationen

I. b. allgemeine Nachteile

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Keine deklarative Programmierung:

- Performance

Ineffiziente Implementierung:

```
vierer(Spielfeld, ...):-
   member(spielstein(F2, Spieler), Spielfeld),
   member(spielstein(F3, Spieler), Spielfeld),
   member(spielstein(F4, Spieler), Spielfeld),
   permutation([F1, F2, F3, F4], [V1, V2, V3, V4]),
   reihe(V1, V2, V3, V4).
```

Effiziente Implementierung:

```
vierer(Spielfeld, ...):-
   permutation([F1, F2, F3, F4], [V1, V2, V3, V4]),
   reihe(V1, V2, V3, V4),
   member(spielstein(F2, Spieler), Spielfeld),
   member(spielstein(F3, Spieler), Spielfeld),
   member(spielstein(F4, Spieler), Spielfeld).
```

I. b. allgemeine Nachteile

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Keine deklarative Programmierung:

- Performance

- Rein deklarative Prolog-Programme sind oft sehr ineffizient im Hinblick auf Speicherverbrauch und Laufzeit

- Dabei sind es nicht technische Aspekte die zur Unbrauchbarkeit führen, sondern vielmehr mathematische Gesetzmäßigkeiten ...

I. b. allgemeine Nachteile

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Keine deklarative Programmierung:

- Performance
- Entnahme von 3 Elementen in geordneter Reihenfolge aus der Liste Spielfeld ohne Wiederholung:

```
vierer(Spielfeld, ...):-
   member(S1, Spielfeld),
   member(S2, Spielfeld),
   member(S3, Spielfeld),
   ...
```

↓

```
Möglichkeiten:
N! / (N - n)!
```

Länge der Liste	Möglichkeiten
3	6
5	60
10	720
20	6840
100	970200

2. a. Vorteile für Spiele KI

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- wenig Verwaltungsoperationen
- Kontrollstrukturen entfallen
- dadurch: kompakter Code, schnelle Entwicklung von Prototypen
- Für Agenten in einem Computerspiel, die logisches Schlussfolgern beherrschen müssen, muss die Inferenz-Engine bei Verwendung von Prolog nicht neu entwickelt werden.

2. b. Nachteile für Spiele KI

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Geringe Unterstützung von APIs
- Wenig Erfahrung bei Entwicklern
- Performance oft das Bottleneck bei Spieleentwicklungen
- Prädikatenlogik oft zu exakt

?- geeignet(prolog).

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Für die Realisierung von KI-Problemen, die logisches Schließen ermöglichen.
- Aber keinesfalls für die Entwicklung eines ganzen Spieles oder der gesamten KI!
- Daher als kleine Programme, die in der Sprache der Entwicklung interpretiert werden.

Agenda

fhwedel
UNIVERSITY OF APPLIED SCIENCES

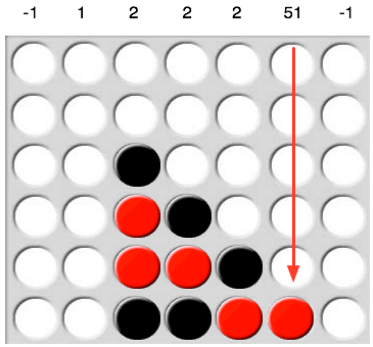
I. Grundlegende Eigenschaften von Prolog

II. Von Aussagen zur Prolog-Syntax

III. Wie Prolog arbeitet

IV. Vor- und Nachteile Prolog

V. Beispiel für Spiele-KI in Prolog



Beispiel für Spiele-KI in Prolog

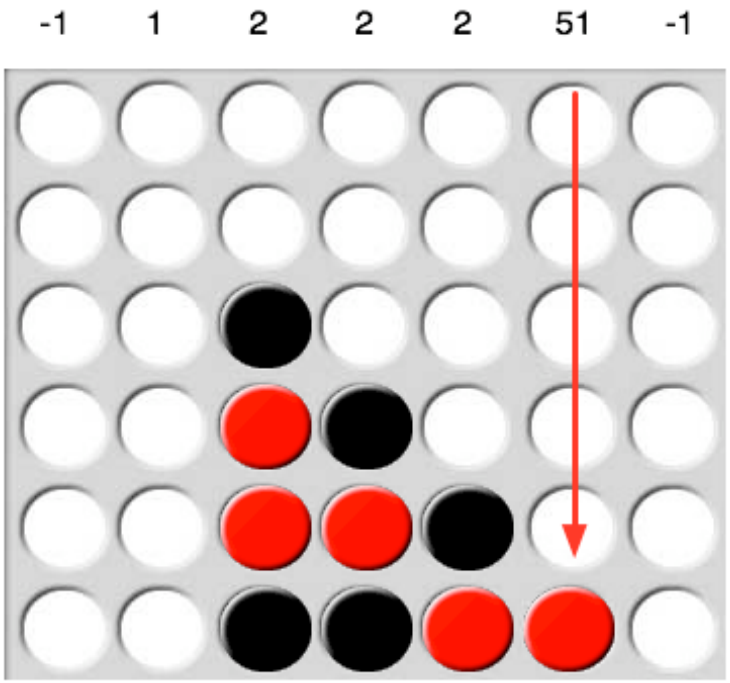
fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Vier Gewinnt: Zwei-Personen-Spiel mit perfekter Information
- Hier als Beispiel für logik-basierte KI
- IO/GUI in Java, KI in Prolog (tuProlog)

Beispiel für Spiele-KI in Prolog

fhwedel
UNIVERSITY OF APPLIED SCIENCES

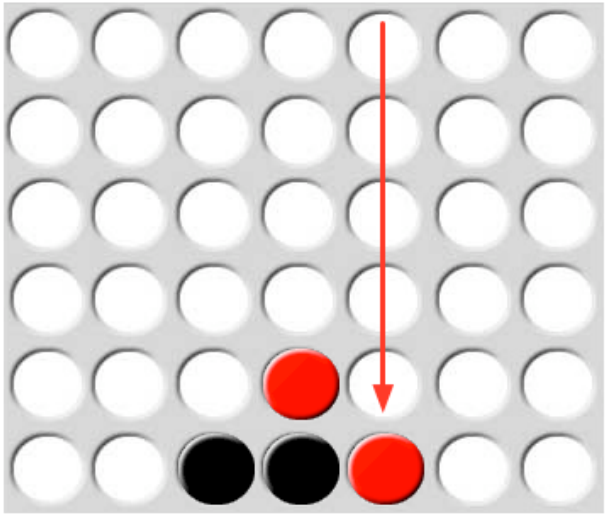
- Zentrale Funktion zug_sum/4



Beispiel für Spiele-KI in Prolog

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Kein Ausprobieren (Alpha-Beta) sondern definieren von guten Spielzügen durch Logik
- Beispiel doppelt offener Dreier



Beispiel für Spiele-KI in Prolog

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Natürliche Sprache:

Es liegt ein doppelter Dreier für ein Spielfeld vor, an dem der Stein $Stein_1$ beteiligt ist, wenn:
- ein auf dem Spielfeld gesetzter Stein $Stein_4$
- mit den Steinen $Stein_1$, $Stein_2$ und $Stein_3$ auf dem Spielfeld eine Vierer-Reihe bildet,
- ein auf dem Spielfeld Spielfeld gesetzter Spielstein $Stein_5$
- ebenfalls mit den Steinen $Stein_1$, $Stein_2$ und $Stein_3$ auf dem Spielfeld eine Vierer-Reihe bildet
- und $Stein_4$ und $Stein_5$ unterschiedliche Steine sind.

Prädikat:

```
doppelter_dreier(Spielfeld, Stein1) :-
   gesetzt(Spielfeld, _, Stein4),
   vierer(Spielfeld, Stein4, Stein1, Stein2, Stein3),
   gesetzt(Spielfeld, _, Stein5),
   vierer(Spielfeld, Stein5, Stein1, Stein2, Stein3),
   Stein4 \= Stein5.
```

Nachvollziehbarkeit?

Effizienz?

Das Ende

fhwedel
UNIVERSITY OF APPLIED SCIENCES

- Vielen Dank für die Aufmerksamkeit!
- Der Quellcode für Vier Gewinnt ist unter

  http://www.mxro.de/

  zu finden