

# Intelligente Systeme

Einleitung, reaktive Agenten, S-R-Agent

Prof. Dr. R. Kruse

C. Moewes

G. Ruß

{kruse,cmoewes,russ}@iws.cs.uni-magdeburg.de

Institut für Wissens- und Sprachverarbeitung

Fakultät für Informatik

Otto-von-Guericke Universität Magdeburg

# Übersicht

## 1. Organisatorisches

Zur Vorlesung

Zur Übung

Zur Prüfung

Inhalt der Vorlesung

Literatur

## 2. Intelligente Systeme

## 3. Reaktive Agenten

## Zu meiner Person: Rudolf Kruse

- ▶ 1979 Dipl. Mathematik (Nebenfach Informatik) von TU Braunschweig
- ▶ dort 1980 promoviert, 1984 habilitiert
- ▶ 2 Jahre hauptamtlicher Mitarbeiter bei Fraunhofer
- ▶ 1986 Ruf als Professor für Informatik der TU Braunschweig
- ▶ seit 1996 Professor an der Universität Magdeburg
- ▶ **Forschung:** Data Mining, Explorative Datenanalyse, Fuzzy-Systeme, Neuronale Netze, EA, Bayes'sche Netze
- ▶ <mailto:kruse@iws.cs.uni-magdeburg.de>
- ▶ Büro: G29-008, Telefon: 0391 67-18706
- ▶ Sprechstunde: Mi., 11:00–12:00 Uhr

# Zur Arbeitsgruppe: Computational Intelligence

Lehre:

- ▶ Intelligente Systeme Bachelor (2 V + 2 Ü, 5 CP)
- ▶ Evolutionäre Algorithmen Bachelor (2 V + 2 Ü, 5 CP)
- ▶ Neuronale Netze Bachelor (2 V + 2 Ü, 5 CP)
- ▶ Fuzzy-Systeme Master (2 V + 2 Ü, 6 CP)
- ▶ Bayes-Netze Master (2 V + 2 Ü, 6 CP)
- ▶ Intelligente Datenanalyse Master (2 V + 2 Ü, 6 CP)
- ▶ (Pro-)Seminare: Information Mining, Computational Intelligence

Forschungsbeispiele:

- ▶ Data Mining in der Landwirtschaft (G. Ruß)
- ▶ Dynamische Graph-Analyse in der Medizin (C. Moewes)
- ▶ Analyse von Sozialen Netzen (P. Held)

# Zur Vorlesung

- ▶ Vorlesungstermine: Do., 9:15–10:45 Uhr, G29-307
- ▶ Vorlesungsende: 26.01.2012
- ▶ Informationen zur Vorlesung:  
<http://fuzzy.cs.ovgu.de/wiki/pmwiki.php?n=Lehre.IS1112>
  - ▶ wöchentliche Vorlesungsfolien als PDF
  - ▶ Übungsblätter ebenfalls
  - ▶ wichtige Ankündigungen und Termine!

# Inhalte der Vorlesung

## Inhalte:

- ▶ Eigenschaften intelligenter Systeme
- ▶ Modellierungstechniken für wissensintensive Anwendungen
- ▶ subsymbolische Lösungsverfahren, heuristische Suchverfahren, lernende Systeme
- ▶ Modellansätze für kognitive Systeme, Wissensrevision und Ontologien
- ▶ entscheidungsunterstützende Systeme
- ▶ weitere aktuelle Methoden für die Entwicklung intelligenter Systeme wie kausale Netze, unscharfes Schließen

# Zur Übung

Lernziele:

- ▶ Befähigung zur Modellierung und Erstellung wissensintensiver Anwendungen durch Auswahl problemementsprechender Modellierungstechniken
- ▶ Anwendung heuristischer Suchverfahren und lernender Systeme zur Bewältigung großer Datenmengen
- ▶ Befähigung zur Entwicklung und Bewertung intelligenter und entscheidungsunterstützender Systeme
- ▶ Bewertung und Anwendung von Modellansätzen zur Entwicklung kognitiver Systeme

Ihre Aufgabe:

- ▶ Nacharbeiten des Vorlesungsstoffs
- ▶ Bearbeitung der Übungsaufgaben
- ▶ aktive Teilnahme an den Übungen

# Durchführung der Übungen

- ▶ Sie werden aktiv und erklären Ihre Lösungen!
- ▶ Tutor macht auf Fehler aufmerksam und beantwortet Fragen
- ▶ das „Vorrechnen“ der Aufgaben ist nicht Sinn der Übung
- ▶ ganz bewusst: keine ausgearbeiteten Musterlösungen
- ▶ Tutoren: Georg Ruß und Christian Moewes
- ▶ `russ@iws.cs.uni-magdeburg.de` bzw.  
`mailto:cmoewes@ovgu.de`
- ▶ Sprechstunde: stets ansprechbar

## Übung: 3 Termine zur Auswahl

- ▶ Di., 15:15–16:45 Uhr in Raum G22B-103
- ▶ Mi., 13:15–14:45 Uhr in Raum G22A-211
- ▶ Mi., 15:15–16:45 Uhr in Raum G22A-211
- ▶ keine Anmeldung erforderlich
- ▶ Übungsbeginn: 18. bzw. 19.10.2011 mit 1. Übungsblatt

# Zur Prüfung

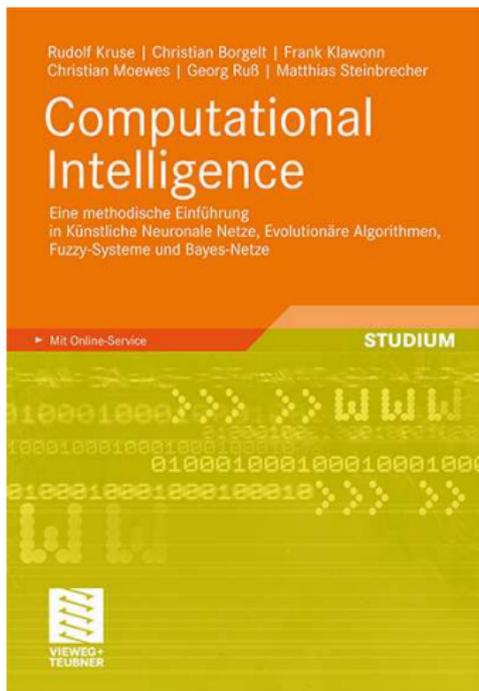
- ▶ schriftliche Klausur: 120 Minuten
- ▶ voraussichtlich im Februar 2012
- ▶ Termine, Räume etc. werden in Vorlesung u. WWW angekündigt
- ▶ alle nichtelektronischen Hilfsmittel zugelassen
- ▶ Bekanntgabe der Ergebnisse: HISQIS
- ▶ Einsichtnahme in die Klausur ist möglich (Termin im WWW)

Schein: Klausur bestehen

# Inhalt der Vorlesung

1. Einleitung, reaktive Agenten, S-R-Agent
2. Neuronale Netze
3. Fuzzy-Systeme
4. Evolutionäre Algorithmen
5. Computational Intelligence
6. Schwarmbasierte Verfahren
7. Zustandsagenten, Problemlösen und Suchen
8. Heuristische Suchalgorithmen
9. Wissensbasierte Systeme, Logik und Prolog
10. Unsicherheit
11. Maschinelles Lernen
12. Fallbasiertes Schließen

# Buch zur Vorlesung



<http://www.computational-intelligence.eu/>

# Literatur zur Lehrveranstaltung I



Beierle, C. and Kern-Isberner, G. (2008).

*Methoden wissensbasierter Systeme : Grundlagen, Algorithmen, Anwendungen.*

Vieweg+Teubner, Wiesbaden, 4., überarbeitete und erweiterte Auflage edition.



Kruse, R., Borgelt, C., Klawonn, F., Moewes, C., Ruß, G., and Steinbrecher, M. (2011).

*Computational Intelligence: Eine methodische Einführung in Künstliche Neuronale Netze, Evolutionäre Algorithmen, Fuzzy-Systeme und Bayes-Netze.*

Vieweg+Teubner-Verlag, Wiesbaden.



Nilsson, N. J. (1998).

*Artificial Intelligence: A New Synthesis.*

Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA.

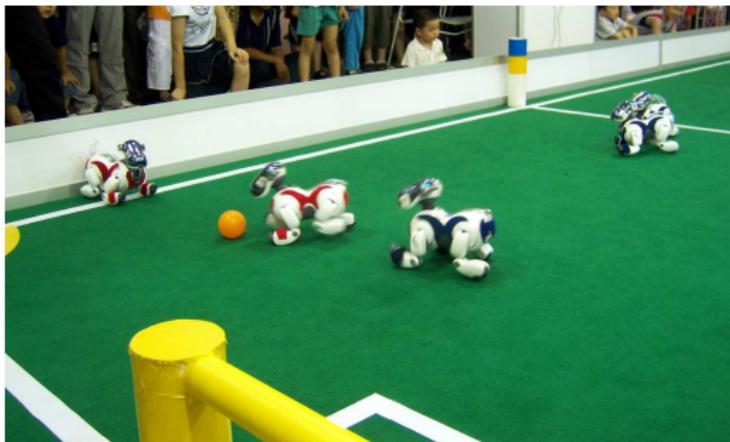
# Übersicht

1. Organisatorisches
- 2. Intelligente Systeme**
3. Reaktive Agenten
4. PAGE-Prinzip
5. Stimulus-Response-Agenten

# Intelligente Systeme beim „Robocup“

Ziel:

*„Bis zum Jahr 2050 soll ein Team von vollständig autonomen humanoiden Robotern entwickelt werden, die gegen das menschliche Fußballweltmeisterschaftsteam gewinnen kann“*



# Humanoider Roboter



weitere Informationen zum Robocup: <http://www.robocup.org>

## ISe bei der „DARPA Grand Challenge“

Ziel:

*„Die DARPA-Stadt-Aufgabe ist ein Forschungs- und Entwicklungsprogramm für autonome Fahrzeuge mit dem Ziel, Technologien zu entwickeln, die mit Kriegsführung und dem Hervorrufen von körperlichen Schäden nichts zu tun haben. Die Stadt-Aufgabe beinhaltet autonome Bodenfahrzeuge, die in einer Pseudo-Stadt manövrieren, simulierte Militärversorgungsmissionen in fließendem Verkehr ausführen, in Kreisverkehren navigieren, volle Kreuzungen passieren und Hindernisse ausweichen.“*

(von <http://www.darpa.mil/grandchallenge/overview.asp>)

# „DARPA Grand Challenge“

Zwei (von elf) Teams: *Stanford Racing* und *Victor Tango*



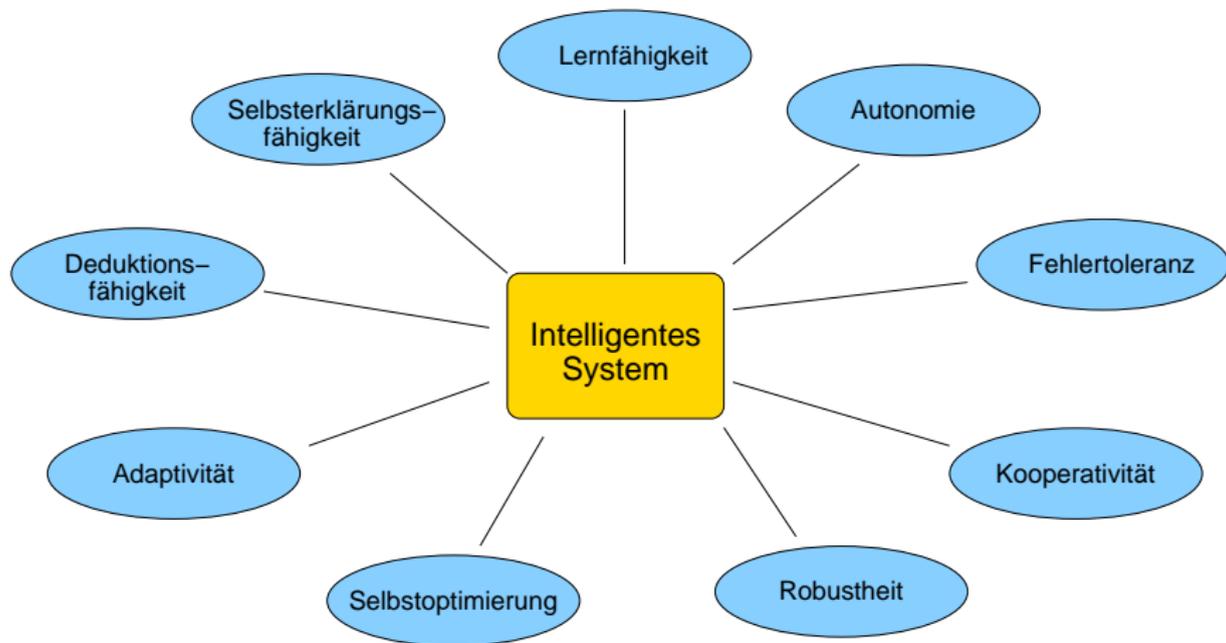
Bild von <http://www.darpa.mil/GRANDCHALLENGE/gallery.asp>

# Was ist ein Intelligentes System

Wissensverarbeitung wird benötigt, um künstliche intelligente Systeme zu entwickeln

- ▶ Begriffsbestimmungen:
  - ▶ *Daten*: Zeichen, die maschinell verarbeitet werden können
  - ▶ *Information*: durch entsprechende Interpretation erhalten Daten einen Sinn; Information ist der abstrakte Inhalt von Daten
  - ▶ *Wissen*: verknüpft Informationen sinngebend miteinander
- ▶ Merkmale intelligenter Systeme:
  - ▶ Lernfähigkeit
  - ▶ Autonomie
  - ▶ Fehlertoleranz
  - ▶ ...

# Merkmale Intelligenter Systeme



# Was ist ein Intelligentes System?

- ▶ Begriff *Intelligentes System* wandelt sich mit wissenschaftlichem Fortschritt:
  - ▶ 1967 Taschenrechner
  - ▶ Programme zur symbolischen Integration
  - ▶ 1997 Schachcomputer DEEP BLUE gewann gegen Kasparov 3.5 zu 2.5
  - ▶ 1998 Automatikgetriebe AG4 für VW New Beetle
  - ▶ 2002 Schachcomputer DEEP FRITZ gegen V. Kramnik 4 zu 4
  - ▶ Expertensysteme (wissensbasierte Systeme, intelligente Systeme)
  - ▶ VW Touareg Stanley bei DARPA Grand Challenge 2006
  - ▶ ...

# Was ist ein Intelligentes System?

- ▶ Problem: oft anmaßende Voraussagen über Fortschritte der Künstlichen Intelligenz (KI)
- ▶ 1957: H. A. Simon (Nobelpreis 1978) und A. Newell (Turing-Award 1975) behaupten, dass 1967
  - ▶ Rechner Schachweltmeister sein wird,
  - ▶ Computer wichtigen neuen mathematischen Satz entdecken und beweisen wird,
  - ▶ digitaler Rechner Musikstück schreiben wird, welchem von Kritikern beachtlicher ästhetischer Wert bescheinigt wird.

# Realistischere Einschätzungen

- ▶ 1993: deutscher Delphi-Report behauptet, dass 2005
  - ▶ Rechner entwickelt werden, die ungenaue Informationen in einer Art gesunden Menschenverstandes verarbeiten können,
  - ▶ Informationsdatenbanken eingesetzt werden, die durch automatisches Lernen ihr Wissen vermehren,
  - ▶ tragbare automatische Übersetzungsgeräte (einfache, alltägliche Konversation in beiden Richtungen) mit Spracheingaben kommerzialisiert werden,
  - ▶ in Büros Geräte weitverbreitet sind, die Texte in handschriftlicher Fließschrift lesen können.

# Was ist ein Intelligentes System?

- ▶ viele Herangehensweisen
- ▶ Gebiet ist hochgradig interdisziplinär
  - ▶ *kognitiver Ansatz*: Simulation kognitiver Prozesse, Analyse menschlicher Denkweise, *general problem solver*
  - ▶ *ingenieurwissenschaftlicher Ansatz*: Konstruktion von Systemen, die gewisse menschliche Wahrnehmungs- und Verstandsleistungen maschinell verfügbar machen (Produkte wie Gesichtserkenner, Roboter, Kooperation mit Gehirnforschern, ...)

# Was ist ein Intelligentes System?

- ▶ spannende philosophische Fragestellungen, wie z.B.  
*Can machines think?*

“can”

- ▶ mehrere Bedeutungen:
- ▶ „Kann denken“: heute – irgendwann – im Prinzip
- ▶ derzeit: Frage „unentscheidbar“, ob Systeme mit menschenähnlichen Fähigkeiten gebaut werden können
- ▶ wir sind mit Fortschritten auf dem Weg dahin zufrieden und verdienen Geld sowie Ruhm mit innovativen Produkten

# Was ist ein Intelligentes System?

*“machine”*

- ▶ muss kein Stahlroboter sein
- ▶ kann auch biologischer Mechanismus sein (Bakterium *Haemophilus influenzae* Rd hat  $10^7$  Basenpaare, 1743 Gene, ...)
- ▶ Was, wenn menschliches Genom entziffert und verstanden?
- ▶ auch: Bewusstseinsdiskussion, Leib-Seele-Problem, das „Ich“ im Gehirn

# Was ist ein Intelligentes System?

“think”

- ▶ Menschen sind Maschinen, also können Maschinen denken
- ▶ Searle (1992): Denken funktioniert nur in speziellen (tatsächlich lebenden) Maschinen
- ▶ Newell & Simon (1976): Physical symbol system (PSS) hypothesis
  - ▶ PSS hat notwendigen und hinreichenden Bedingungen für intelligentes Verhalten
  - ▶ PSS ist eine Maschine, die symbolische Daten manipulieren kann (z.B. Computer)
- ▶ Kohonen u.a. (1980): Entwicklung intelligenter Maschinen nur durch subsymbolische Prozesse (z.B. Signale)
- ▶ Zadeh (1964): wirklich intelligente Systeme müssen Art *fuzzy logic* benutzen (keine binäre Logik)
- ▶ Turing-Test (1950)

## Turing-Test (1950)

Der Turing-Test wird von drei Personen gespielt: einem Mann (A), einer Frau (B) und einem Fragesteller (C). Der Fragesteller befindet sich in einem Raum, abgeschottet von A/B, und kommuniziert mit diesen über ein Terminal (teletype). Das Ziel des Spiels für den Fragesteller ist zu bestimmen, welche der beiden Personen der Mann und welche die Frau ist. Er adressiert die beiden mit Variablen X/Y und am Ende des Spiels sagt er "X ist A und Y ist B" oder "X ist B und Y ist A". Der Fragesteller kann Fragen beispielsweise der folgenden Form stellen:

*C: X, würden Sie mir bitte Ihre Haarlänge verraten?*

Wenn mit "X" A adressiert wird, dann muß A jetzt antworten. Für A geht es darum, C in die Irre zu leiten und ihn zu einer falschen Identifikation zu verleiten.

## Turing-Test (1950)

Das Ziel für den dritten Spieler B ist, dem Fragesteller zu helfen.

...

Jetzt stellen wir uns die Frage: “Was passiert, wenn eine Maschine den Anteil von A an diesem Spiel übernimmt?” Wird sich der Fragesteller genauso oft falsch entscheiden wenn das Spiel mit einer Maschine gespielt wird, wie wenn es mit Mann/Frau gespielt wird? Diese Fragestellung ersetzt das ursprüngliche “Kann eine Maschine denken?”

Der Turing-Test wird oft vereinfacht zu einem Test, in dem eine Maschine versucht, einen menschlichen Fragesteller dazu zu verleiten, sie als Mensch zu identifizieren.

# Was ist ein Intelligentes System?

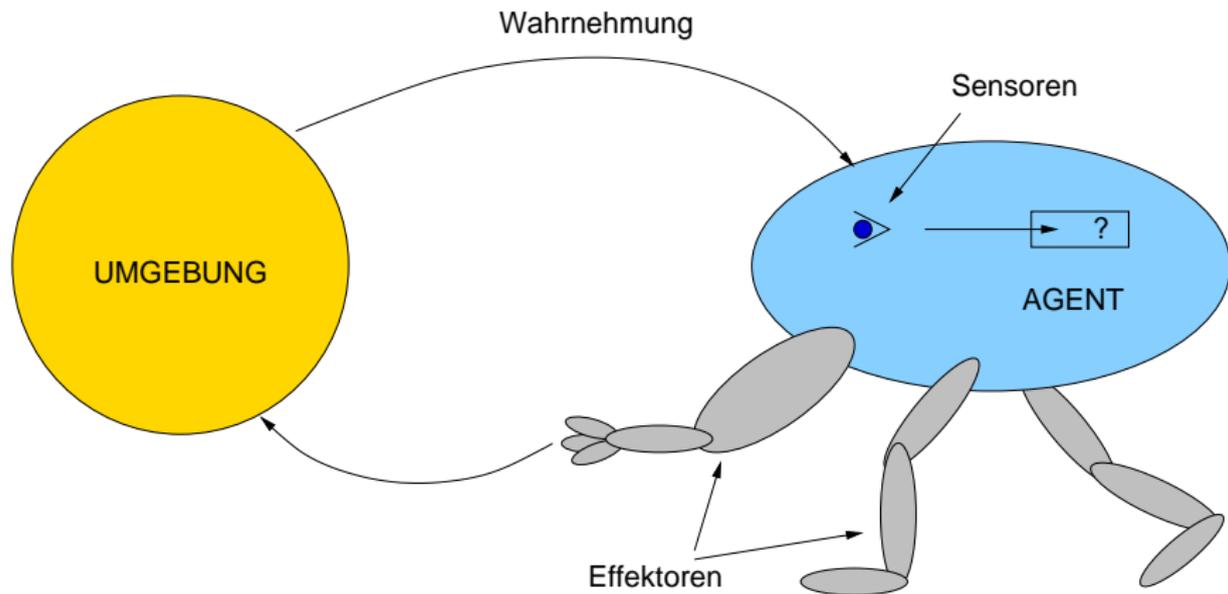
- ▶ Gebiet der Wissensverarbeitung ist extrem innovativ:
  - ▶ objektorientierte Programmiersprachen
  - ▶ graphische Oberflächen
  - ▶ Expertensysteme
  - ▶ Software-Agenten (Internet)
  - ▶ Autonome Robotersind hier erfunden worden.

# Übersicht

1. Organisatorisches
2. Intelligente Systeme
- 3. Reaktive Agenten**
4. PAGE-Prinzip
5. Stimulus-Response-Agenten

# Agenten

Ein intelligenter Agent interagiert mit seiner Umgebung mittels Sensoren und Effektoren und verfolgt gewisse Ziele:



# Beispiele für Agenten

- ▶ Menschen und Tiere
- ▶ Roboter und Software-Agenten (Softbots)
- ▶ aber auch: Heizungen, ABS, ...



# Agenten

auch andere Definitionen aus unterschiedlichen Fachgebieten, z.B.:

Ein Programm ist ein Softwareagent, wenn es korrekt in einer (Agenten-)Sprache wie ACL, KQML oder KIF kommuniziert. BDI-Agenten werden durch Überzeugungen (*beliefs*), Wünsche (*desires*) und Absichten (*intentions*) beschrieben; praktisch werden sie mit einer Modallogik und speziellen Datenstrukturen implementiert.

# Agenten

- ▶ Beispiel: Simulation Soccer
- ▶ RoboCup: Roboterfußball



## Beispiel 2: Taxifahrer

Typ	Taxifahrer
Wahrnehmung	Kameras, Tachometer, GPS, Mikrofon
Aktionen	Steuern, Schalten, Bremsen, mit Fahrgästen sprechen
Ziele	Sichere, schnelle, legale, komfortable Fahrt; Profit maximieren
Umgebung	Straßen, andere Verkehrsteilnehmer: Fußgänger, Radfahrer; Fahrgäste

# Übersicht

1. Organisatorisches
2. Intelligente Systeme
3. Reaktive Agenten
- 4. PAGE-Prinzip**
5. Stimulus-Response-Agenten

# Charakterisierung von Agenten

- ▶ Agenten können charakterisiert werden durch:
  - ▶ Wahrnehmungen (*perceptions*)
  - ▶ Aktionen (*actions*)
  - ▶ Ziele (*goals*)
  - ▶ Umgebung (*environment*)

⇒ *PAGE*

## Beispiele von Agenten nach *PAGE*

Art	Wahrnehmung	Aktionen	Ziele	Umgebung
Medizinisches Diagnosesystem	Symptome, Diagnose, Antworten des Patienten	Fragen, Tests, Behandlungen	Gesundheit, geringe Kosten	Patient, Krankenhaus
Satellitenbildanalyse	Punkte verschiedener Intensität	Klassifikation	Korrekte Klassifikation	Satellitenbilder
Roboter	Punkte verschiedener Intensität	Teile aufheben und einsortieren	Teile richtig einsortieren	Förderband mit Teilen

## Beispiele von Agenten nach *PAGE*

Art	Wahrnehmung	Aktionen	Ziele	Umgebung
Raffinerie-Regler	Temperatur, Druck	Öffnen, Schließen von Ventilen, Temperatur einstellen	Reinheit, Ertrag, Sicherheit maximieren	Raffinerie
Interaktiver Englisch-Tutor	Eingegebene Wörter und Übungen, Vorschläge	Korrekturen ausgeben	Testergebnisse des Studenten maximieren	Menge von Studenten

# Typen von Agenten (I)

Unterscheidung von Agenten nach Art und Weise ihrer Umwelt-Interaktionen:

- ▶ *reaktive Agenten:*
  - ▶ steuern über ein Reiz-Antwort-Schema ihr Verhalten
- ▶ *reflektive Agenten:*
  - ▶ agieren planbasiert, verarbeiten also explizit Pläne, Ziele und Intentionen
- ▶ *situierte Agenten:*
  - ▶ verbinden einfaches Reagieren und überlegtes Handeln in dynamischer Umwelt

## Typen von Agenten (II)

- ▶ *autonome Agenten:*
  - ▶ sind zwischen reflektiven und situierten Agenten einzuordnen (werden meist in Robotik verwendet)
- ▶ *rationale Agenten:*
  - ▶ entsprechen reflektiven Agenten, allerdings mit ausgeprägter Bewertungsfunktionalität
- ▶ *soziale Agenten:*
  - ▶ sind in der Lage, ihr Handeln an Gemeinziel auszurichten

# Übersicht

1. Organisatorisches
2. Intelligente Systeme
3. Reaktive Agenten
4. PAGE-Prinzip
- 5. Stimulus-Response-Agenten**

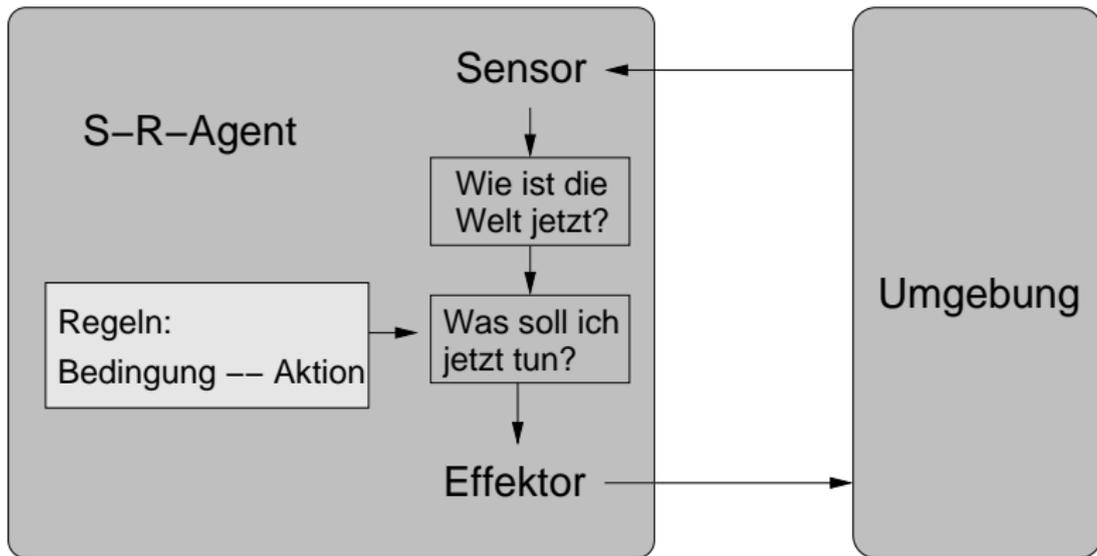
Gitterwelt

Signalverarbeitung

Distributed World of Agents

# Stimulus-Response-Agent

einfacher reaktiver Agent: antwortet unmittelbar auf Wahrnehmungen



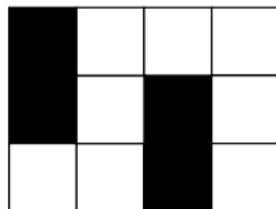
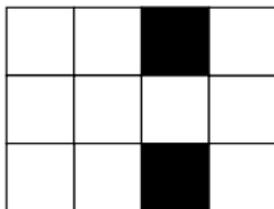
## „Skelett“ eines Stimulus-Response-Agenten

```
function S-R-Agent(percept) returns action
  static: rules
  state INTERPRET-INPUT(percept)
  rule RULE-MATCH(state, rule)
  action RULE-ACTION(rule)
  return action
```

1. Agent sucht Regel, deren Bedingung der gegebenen Situation entspricht
2. er führt zugehörige Aktion (Regel-Konklusion) aus

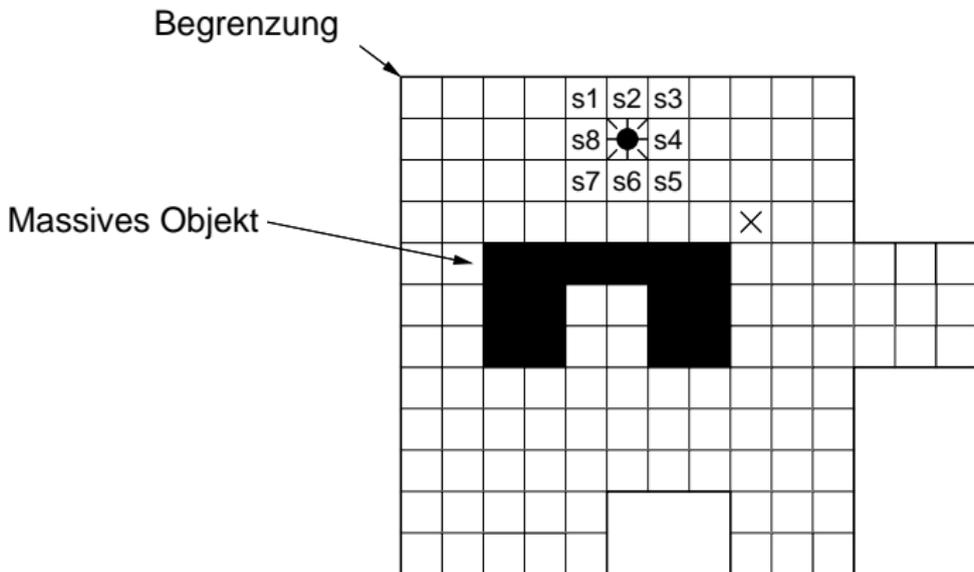
## Gitterwelt (I)

- ▶ Umwelt = fiktive zweidimensionale Gitterzelleneinheit
- ▶ verschiedene (Spielzeug-)Agenten tummeln sich dort
- ▶ in Zellen können Objekte mit verschiedenen Eigenschaften sein
- ▶ es gibt Barrieren
- ▶ Agenten können von Zelle zu Zelle laufen
- ▶ keine engen Zwischenräume, d.h. keine Lücken zwischen Objekten und Begrenzungen, die nur 1 Zelle breit sind (*tight spaces*)



Solche Umgebungen sind *nicht* erlaubt!

## Gitterwelt (II)

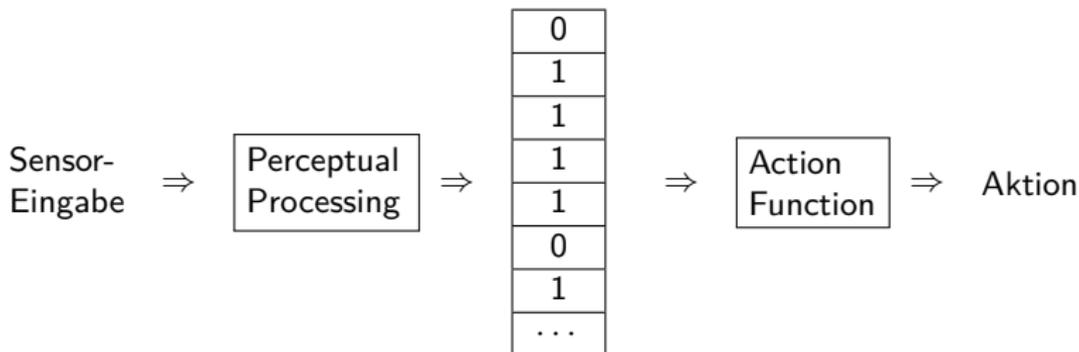


- ▶ Roboter kann mithilfe der Sensoren  $s_1, \dots, s_8$  feststellen, welche Zellen in seiner Nachbarschaft belegt sind

## Gitterwelt (III)

- ▶  $s_j \in \{0, 1\}$ , Sensoreingabe:
  - ▶  $s_j = 0 \Leftrightarrow$  Zelle  $s_j$  ist frei für Roboter
  - ▶ an der mit  $\times$  markierten Stelle: (0, 0, 0, 0, 0, 0, 1, 0)
- ▶ vier mögliche Aktionen:
  - ▶ north, east, south, west
  - ▶ z.B. north bewegt Roboter 1 Zelle nach oben, falls Zelle frei ist, ansonsten wird nicht bewegt
- ▶ Aufgabe häufig in 2 Schritten gelöst:
  1. Phase: perception processing
  2. Phase: action computation

## Komponenten: *Perception* und *Action*



- ▶ Eigenschaftsvektor  $X = (0, 1, 1, 1, 1, 0, 1, \dots)^T$
- ▶ vom Entwickler zugewiesene Bedeutungen:
  - ▶  $(0, 1, \mathbf{1}, 1, 1, 1, \dots)$ : „an einer Wand“
  - ▶  $(0, 1, 1, 1, \mathbf{1}, 1, \dots)$ : „in einer Ecke“

## Beispiel: Wandverfolgung (I)

- ▶ Aufgabe: gehe zu einer Zelle an Begrenzung eines Objekts und folge dieser Grenze
- ▶ Perception:
  - ▶  $2^8$  verschiedene Sensoreingaben, von denen einige wegen Einschränkung (*keine engen Zwischenräume*) wegfallen
  - ▶ vier Merkmale  $x_1, \dots, x_4$ :

$$x_1 = 1 \Leftrightarrow (s_2 = 1 \vee s_3 = 1)$$

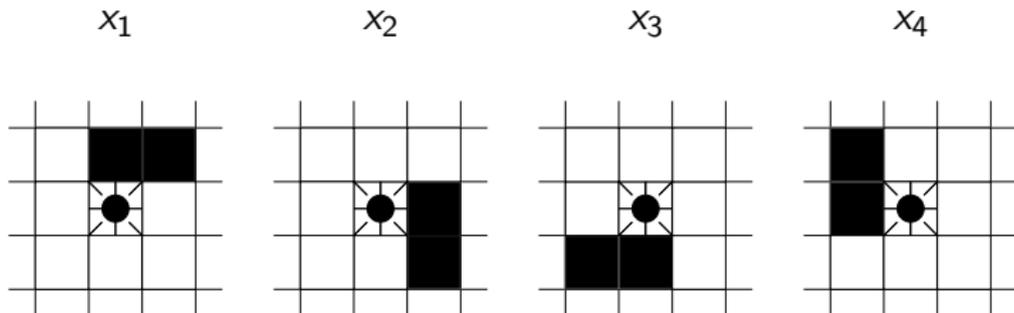
$$x_2 = 1 \Leftrightarrow (s_4 = 1 \vee s_5 = 1)$$

$$x_3 = 1 \Leftrightarrow (s_6 = 1 \vee s_7 = 1)$$

$$x_4 = 1 \Leftrightarrow (s_8 = 1 \vee s_1 = 1)$$

## Beispiel: Wandverfolgung (II)

- ▶ Merkmal in jedem Diagramm hat genau dann Wert 1, wenn mindestens 1 der markierten Zellen belegt



- ▶ in komplexen Welten: Informationen sind typischerweise unsicher, vage, oder sogar falsch

## Beispiel: Wandverfolgung (III)

Aktionen:

- ▶ falls keines der 4 Merkmale Wert 1 hat, führe `north` durch
- ▶ sonst:

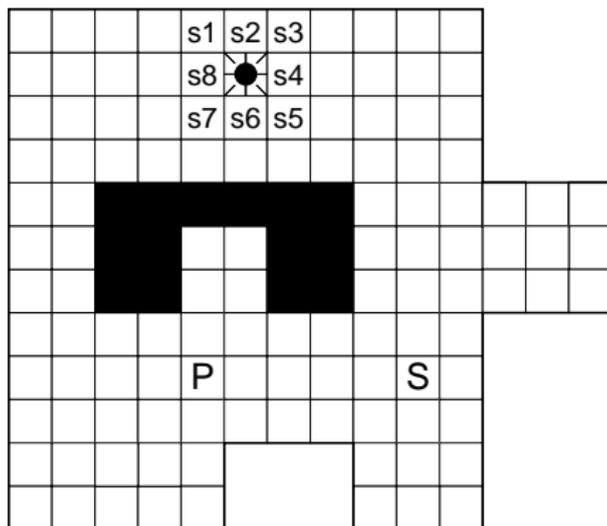
If  $x_1 = 1$  and  $x_2 = 0$  then `east`

If  $x_2 = 1$  and  $x_3 = 0$  then `south`

If  $x_3 = 1$  and  $x_4 = 0$  then `west`

If  $x_4 = 1$  and  $x_1 = 0$  then `north`

## Beispiel: Wandverfolgung (IV)



- ▶ Roboter, der an Position  $P$  startet, bewegt sich entgegen Uhrzeigersinn am Objekt entlang
- ▶ Roboter, der an Position  $S$  startet, bewegt sich im Uhrzeigersinn an der äußeren Begrenzung entlang

# Auswertung der Sensoreingaben

- ▶ für beiden Phasen *perception processing* und *action computation* werden oft Boolesche Algebren verwendet
- ▶ so gilt:

$$\begin{aligned}x_4 = s_1 \vee s_8 \text{ und go north} \\ \Leftrightarrow \\ (\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge \bar{x}_4) \vee (x_4 \wedge \bar{x}_1) = 1\end{aligned}$$

## Auswertung der Sensoreingaben

- ▶ geeignete Repräsentationsform für Aktionen sind Regelsysteme der Form  $c_j \rightarrow a_j$ , wobei
  - ▶  $c_j$  der Bedingungsteil und
  - ▶  $a_j$  der Aktionsteil sind
- ▶ in unserem Beispiel erhält man folgende Regeln:

$$x_4 \wedge \bar{x}_1 \rightarrow \text{go north}$$

$$x_1 \wedge \bar{x}_2 \rightarrow \text{go east}$$

$$x_2 \wedge \bar{x}_3 \rightarrow \text{go south}$$

$$x_3 \wedge \bar{x}_4 \rightarrow \text{go west}$$

$$1 \rightarrow \text{go north}$$

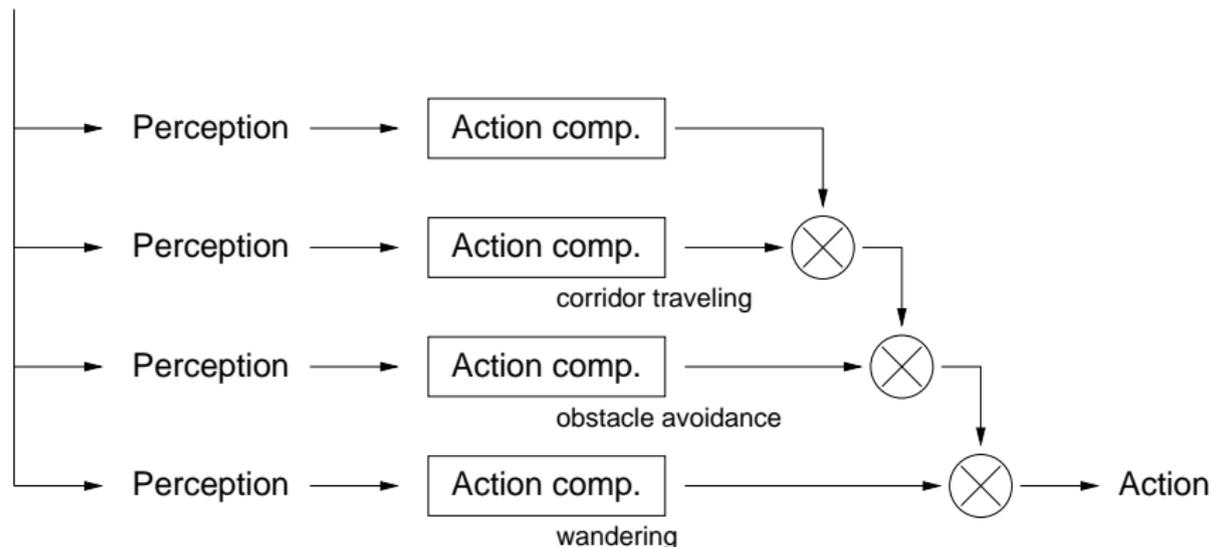
- ▶ Regelsysteme und Boolesche Algebren kann man gut anhand von Netzwerken implementieren

# Wandverfolgung

- ▶ alternativer Ansatz zu S-R-Agenten: Einführung von Subsumptions-Modulen
  - ▶ jedes Modul enthält Sensorinformationen direkt von Umwelt
  - ▶ sind spezifizierten Voraussetzungen des Moduls erfüllt, wird Programm ausgeführt
  - ▶ „höhere“ Module subsumieren „tiefere“, d.h. falls Voraussetzung eines höheren Moduls erfüllt, wird tieferes Modul durch höheres ersetzt

# Wandverfolgung

Sensor Signals



# Sicherheitskritische Systeme

- ▶ Fehlfunktion kann zu schweren Unfällen, Umwelt- oder physischen Schäden oder Todesopfern führen
- ▶ Fehlentscheidungen solcher Systeme können für gewöhnlich nicht korrigiert werden



## Airbag-Zündung

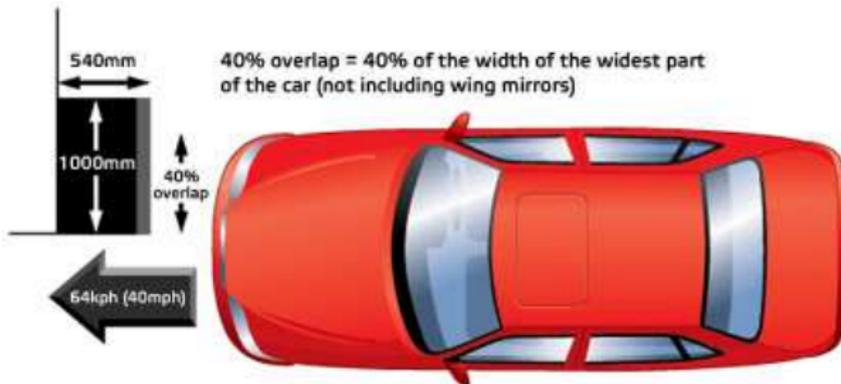
Airbag soll nur in schweren Crashes zünden



## Krankheitsdiagnose

kein kranker Patient soll als gesund diagnostiziert werden

# Frontal-Crashes



Quelle: <http://www.euroncap.com/tests/frontimpact.aspx>

- ▶ gegen starre Wand (engl. wall crash)
- ▶ mit relativem Winkel von  $30^\circ$  zwischen 2 Autos (angular crash),
- ▶ gegen deformierbare Barriere mit Versatz (offset deformable barrier (ODB) crash)
- ▶ bei  $v = 15 \text{ km/h}$  (Allianz Zentrum für Technik (AZT) crash)

# Crash-Sensorik



Airbag-  
steuergerät



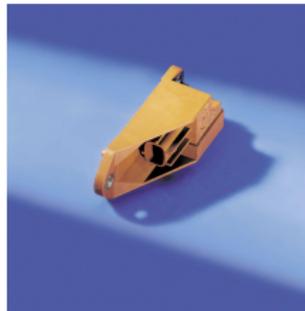
Sensor-  
positionen



Insassen-  
klassifikations-  
system



“Early Crash”  
Sensor

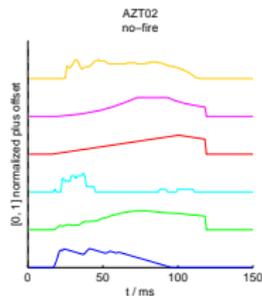
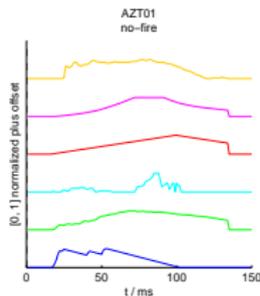
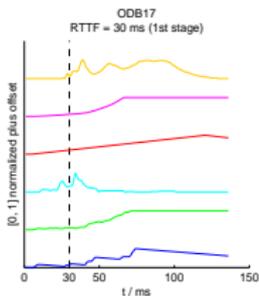
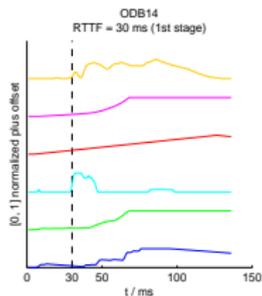
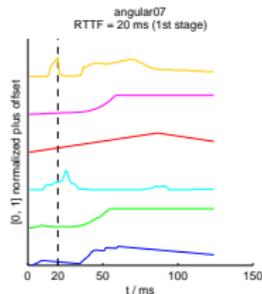
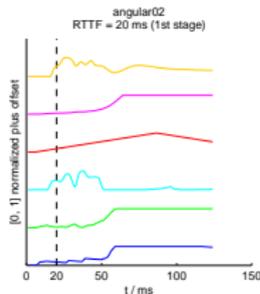
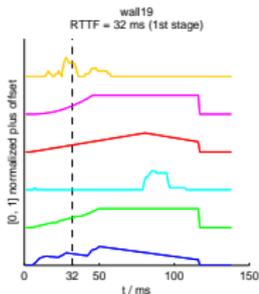
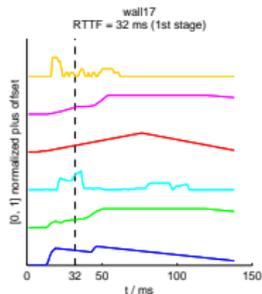


Beschleunigungs-  
sensor



Drucksensor

# Beispielhafte Crash-Signale



# Stand der Technik

- ▶ Verwendung von regelbasierten Agenten (**manuell erstellt**)
- ▶ Zwei-Klassen-Problem: positive (Crash, krank) und negative (kein Crash, gesund) Beispiele (Fahrten, Patienten)
- ▶ jedes Beispiel: beschrieben durch numerischen Attributwerten

## Herausforderungen:

- ▶ **automatisches Lernen**
- ▶ Dutzende von Attributen ( $\geq 50$  beim Airbag)
- ▶ wenige Beispiele aufgrund von hohen Experimentkosten
- ▶ Falsch-Positive (Fehlalarme) müssen verhindert werden
- ▶ von Experten gewünscht: verständliche, regelbasierte Agenten

# Reale Regelbasis: Erkennen von Wand-Crashes

- 1: (X01 >= 221) & (X02 >= 164) => deploy airbag
- 2: (X01 >= 215) & (X03 >= 22) => deploy airbag
- 3: (X04 >= 225) & (X02 >= 188) => deploy airbag
- 4: (X01 >= 177) & (X02 >= 248) => deploy airbag
- 5: (X04 >= 176) & (X02 >= 236) => deploy airbag
- 6: (X04 >= 171) & (X02 >= 231) & (X05 >= 74) & (X06 >= 14) => deploy airbag
- 7: (X04 >= 164) & (X07 >= 3) & (X06 >= 9) => deploy airbag
- 8: (X02 >= 224) & (X03 >= 26) => deploy airbag
- 9: (X01 >= 149) & (X08 >= 106) => deploy airbag
- 10: (X01 >= 144) & (X03 >= 32) => deploy airbag
- 11: (X04 >= 150) & (X07 >= 4) => deploy airbag
- 12: (X07 >= 2) & (X02 >= 255) => deploy airbag
- 13: (X07 >= 5) & (X09 >= 131) => deploy airbag
- 14: (X010 >= 255) & (X07 >= 3) & (X05 >= 231) => deploy airbag
- 15: (X010 >= 255) & (X07 >= 3) & (X011 >= 77) & (X06 >= 10) => deploy airbag
- 16: (X07 >= 3) & (X06 >= 24) & (X09 >= 134) => deploy airbag
- 17: (X010 >= 255) & (X07 >= 2) & (X02 >= 50) & (X03 >= 22) => deploy airbag
- 18: (X010 >= 255) & (X07 >= 2) & (X02 >= 188) & (X03 >= 11) => deploy airbag
- 19: (X010 >= 255) & (X07 >= 2) & (X02 >= 188) & (X05 >= 255) & (X06 >= 26) => deploy airbag
- 20: (X07 >= 2) & (X02 >= 179) & (X03 >= 14) => deploy airbag
- 21: (X07 >= 2) & (X02 >= 176) & (X08 >= 90) => deploy airbag
- 22: (X010 >= 255) & (X07 >= 2) & (X01 >= 128) & (X08 >= 93) & (X06 >= 15) => deploy airbag
- 23: (X010 >= 255) & (X07 >= 1) & (X01 >= 131) & (X08 >= 103) & (X06 >= 23) => deploy airbag
- 24: (X07 >= 2) & (X01 >= 133) & (X08 >= 137) => deploy airbag
- 25: (X01 >= 131) & (X08 >= 105) & (X03 >= 24) => deploy airbag
- 26: (X010 >= 226) & (X07 >= 3) & (X012 >= 13) & (X06 >= 15) & (X03 >= 3) => deploy airbag
- 27: (X07 >= 2) & (X01 >= 115) & (X06 >= 52) & (X03 >= 19) => deploy airbag
- 28: (X010 >= 138) & (X07 >= 3) & (X01 >= 94) & (X03 >= 9) & (X09 >= 113) => deploy airbag

# Erlernen der Agenten aus Crash-Daten

- ▶ Eingabe: 50-dimensionale numerische Crash-Daten
- ▶ Verarbeitung: durch robustes Lernverfahren
- ▶ Ausgabe: leicht verständlicher Agent als „kompakte“ Regelmenge

## Vorteile:

- ▶ jede Regel: besteht aus meistens 3 (max. 5) von 50 Attributen
  - ▶ nahezu alle Crashes werden **rechtzeitig** erkannt
  - ▶ Falsch-Positive  $\approx 0\%$ , Falsch-Negative  $\approx 25\%$
- ⇒ keinerlei Fehlzündungen
- ▶ Falschnegative sind nicht so wichtig, da Crashes auch im späteren Verlauf erkannt werden können