Hochschule Bremerhaven

Lecture Notes

Agentensysteme (WP 43)

Wintersemester 2022/23

Version: 13. Juli 2023

Prof. Dr. Lars Fischer

lars.fischer@hs-bremerhaven.de

License

Contact me via email at lars.fischer@hs-bremerhaven.de.

Inhaltsverzeichnis

0

Syllabus

> Den besten Teil unserer Lebenszeit verbringen wir auf der Arbeitsstelle. Man muss deshalb lernen, so zu arbeiten, dass die Arbeit leicht und zu einer ständigen Lebensschule wird.

Für alle Probleme und Lernhindernisse gilt eine fundamentale Regel:

> Probleme müssen rechtzeitig **vor** dem Start- bzw. Abgabetermin einer einzelnen Prüfungsleistung mitgeteilt werden.

Mit dem Start (beziehungsweise der Abgabe) einer Prüfungsleistung ändert sich der Arbeitsmodus von "Lernen und Lehren" zu "Prüfen". Im Modus "Lernen und Lehren" möchte ich Ihnen jede Unterstützung angedeihen lassen die möglich ist um ihren Lernerfolg zu fördern. Im Modus "Prüfen" ist es meine Aufgabe diesen Lernerfolg zu prüfen und zu bewerten und damit auch den Wert eines erfolgreichen Kursabschlusses sicherzustellen. Praktisch bedeutet dies, dass es nach dem Start einer Prüfungsleistung (beziehungsweise dem Abgabezeitpunkt von Übungsaufgaben) keine Entgegenkommen bezüglich Terminen, Prüfungsinhalten oder sonstige Anpassungen der Lehre und Prüfung geben kann. Natürlich stehen ihnen alle vorgesehenen formalen Wege, wie die Einreichung von ärztlichen Attesten oder die Anrufung des Prüfungsausschusses offen.

Im Anschluß an die Prüfungsphase haben Sie aber immer die Gelegenheit die Bewertung ihrer Prüfungsleistung einzusehen und die Korrektur von Bewertungsfehlern einzufordern.

Lern- und Prüfungsphasen

Abbildung 1: Lern-, Lehr- und Prüfungsphasen

Rahmenbedingungen

Rahmenbedingungen

Präsenzzeit: 2 VL, 2 Ü (3h 20m)

Selbstlernzeit: (4h 40m)

Semesterzeiten:
- Vorlesungsstart: 2023-04-11
- Vorlesungsende: 2023-07-14

Zielgruppe:

Studiengänge: INF und WINF

- Fachsemester: (Wahlpflicht)

Prüfungsform: Entwurf

Lernziele

Lernziele

- Anwendbare Kenntnis des Agentenmodells auf unterschiedliche Probleme
- Kenntnis unterschiedlicher Agententechnologien

Prüfungsleistung

Die Prüfungsform ist der Entwurf

> ". Ein Entwurf ist die Erstellung eines Designs bzw. eines Modells und/oder einer Implementierung, die mit fachspezifischen Methoden entwickelt wird. Er kann auch in einer Gruppenarbeit erstellt werden." [§3 (1) BPO HBv INF]

Der Entwurf in WP 43 Agentensysteme umfasst

Prüfungsleistung

1. lauffähiger Prototyp
   - vollständiger Quellcode am Stichtag über gitlab verfügbar
   - Funktionsumfang nach Aufgabenstellung
2. Dokumentation (10 Seiten)
   - Konzept der Implementation
   - Ein Vertiefungsthema nach Absprache
3. Abschlußpräsentation (10 Minuten)
4. Zwischenstandpräsentation (10 Minuten)

Zeitplan

Übungen Zwischenpräsentationen

15.7. Präsentation Entwurf (praktischer Teil)

Quellen

- **russell2010artificial**: **russell2010artificial**; **russell2010artificial** [**russell2010artificial**]
- **BraunRossak05**: **BraunRossak05**; **BraunRossak05** [**BraunRossak05**]

1

Agents

Lernziele

- Das Konzept "Agent"
- Sensoren, Actuatoren und Percepts
- Begriff Environment im Agentenmodel
- Klassifizierung von Problemen mittes PEAS

Agents

> "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators." [**russell2010artificial**]
>
> "For each possible percept sequence, a rational agent should select an action expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has." [**russell2010artificial**]

Agent-Environment-Interaction

Examples

"Agent System" is a model for problems.

Can you imagine a few fitting problems?

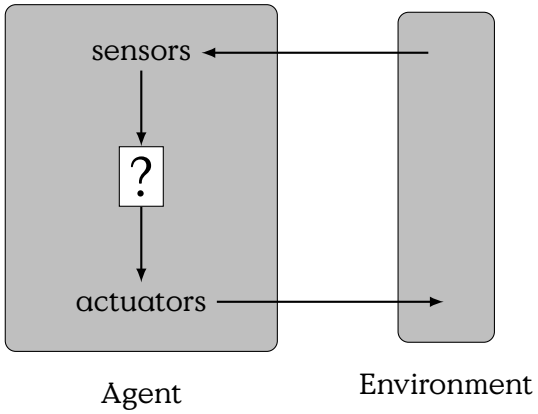


Figure 1.1: [**russell2010artificial**]

Percept Sequence

Percept Sensor-Input zu einem bestimmten Zeitraum

Percept sequence Historie aller percepts eines Agenten

Agent Features

Minimal features of (Wooldridge-Jennings) agents [**BraunRossak05**]:

Autonomy operate and behave, self-made plan

Social behavior agent communication language

Reactivity perceive their environment by sensors and react to identified events

Proactivity Goal driven behaviour

Examples

Agent

Non-Agent

1.1 Rational Agents

Rational Agents

> For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure[...]. [**russell2010artificial**]

Bsp. Fußballroboter

rationality: maximizes expected performance

omniscience: "knows" the actual outcome of actions agent

autonomy: agent is able to falsify/extend prior knowledge

Rational vs. Perfect

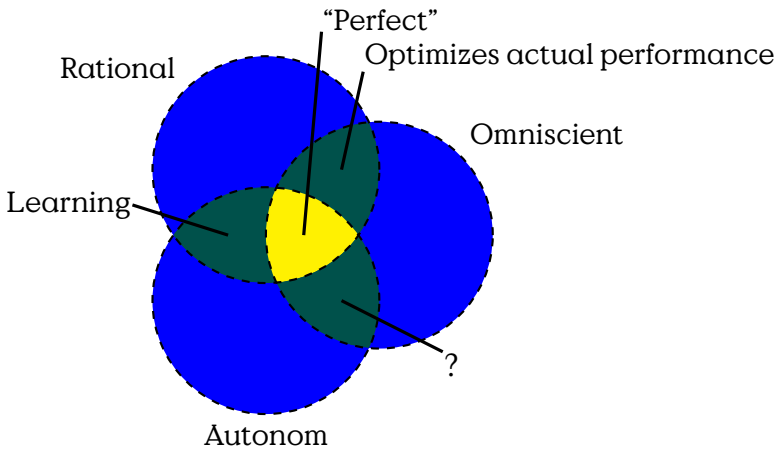


Abbildung 1.2: Rationality, omniscience and autonomy

Rational Agent

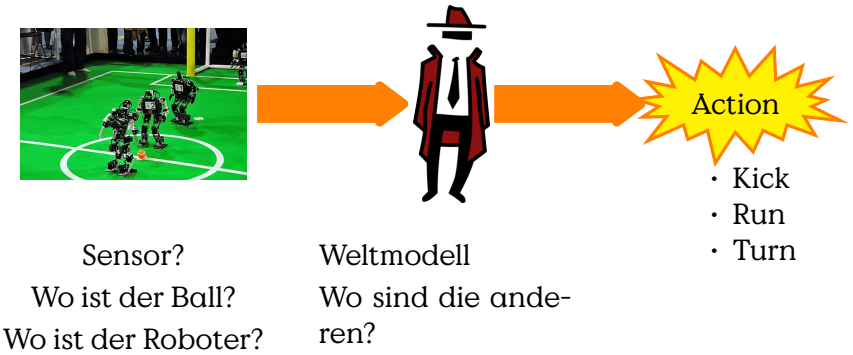


Abbildung 1.3: Example: Football agents

Umfrage:

- Ein Fußballroboter fährt zufällig über das Spielfeld.
- Sobald er einen Ball erkennt, führt er die Aktion “Schuss” aus.
- Der Roboter nutzt also:
  1. Percept
  2. Percept Sequence

1.2 Task Environment

The task environment denominates the “problem” that agents can be a solution for. In here we describe it more formally as the performance, environment, actuators, sensors that are available.

The task environment is as crucial to an agent system as the agents themselves. It defines not only the available tools and perceptions that can be used by agents, but limits and rules of the environment and even the definition of success. In other word:

PEAS

Performance measure of success for the agent, this determines rationality for the particular problem

Environment everything an agent may face, other agents, passive elements, the “area” or other degrees of freedom where an agent may “move”.

Actuators define the ways in which an agent can interact with its environment in order to change states of itself or other entities.

Sensors the interfaces for information input, i. e., the “eyes” and “ears” of an agent.

Self-Driving Taxi Example

1.3 Agent Types

Agent Types

Observability fully vs. partially: For a given point in time, can the agent observe the complete state of an environment or just a part of it?

Performance	Environment	Actuators	Sensors
Safe, fast, legal, comfort, maximize profit, minimize impact to others	Roads, other traffic, police, pedestrians, customers, weather	steering wheel, Brake, accelerator, signal horn	Cameras, rader, speedometer, GPS, engine sensors, accelerometer, microphones, ...

Table 1.1: Example: Self-driving Taxi

Task Env.	Observable	#Agents	Det.	Episode	Static	Diskret
Sudoku	Fully	Single	Det.	Seq.	Static	Discrete
Schach	Fully	Multi	Det.	Seq.	Semi	Discrete
Bildanalyse	Fully	Single	Det.	Episodic	Semi	Continuous
Taxifahren	Fully	Multi	Det.	Seq.	Semi	Discrete
Sortierroboter	Partial	Single	Stoch.	Episodic.	Dynamic	Continuous
Englishlehrer	Partial	Multi	Stoch.	Seq.	Dyn.	Discrete
<Dein Beispiel>						

Tabelle 1.2: Beispiele Task Environment [**russell2010artificial**]

Multi-/Single-Agent is the problem comprised of only a single agent or does the agent have to consider other entities to be agents as well?

Competitive vs. cooperative

Determinism deterministic, nondeterministic, stochastic

Episodic vs. sequential single perception vs. percept sequences

Static vs. dynamic i. e., can the environment change concurrently. If the environment changes while the agent is deliberating, than it is dynamic. If only the performance score is changing, we could call it semistatic.

Discrete or contiuous

Known/Unknown environmental laws and rules

Beispiele

siehe Tabelle **??**.

1.4 Framework

1.4.1 Python

Wir nutzen in diesem Kurs das Agentenframework `mango` zur Implementierung unserer Beispiele. Dafür ist es notwendig sich ein wenig in Python einzuarbeiten. Wir werden das anhand eines Tutorials und wenigen weiteren Quellen machen, die wir nach und nach in den Übungen und den Hausaufgaben durcharbeiten.

Die Grundelemente lassen sich in der Infrastruktur der Informatik ausprobieren.

Python Referenzen

- Python Basics
- Entwicklungsumgebungen
- Python Styleguide
- Mango Agenten Tutorial

Der erste Schritt in der Entwicklung ist das anlegen einer eigenen Umgebung in der alle Bibliotheken, Interpreter und deine Skripte zu liegen kommen. Um deinen Code in deiner Entwicklungsumgebung ausführen kannst, musst du die Umgebung aktivieren.

Python Environment Setup

```
mkdir projekt
python3 -m venv projekt/env
source projekd/env/bin/activate
pip install mango-agents
```

(`projekt` ist dein Projektname)

Mit `pip` installierst du Pakete aus den Repositories bei `python.org`. Wir benötigen zuerst nur die Bibliothek mit dem Agentenframework `mango-agents`.

Weitere Agentenplattformen:

- Python Intelligent Agent Framework (piaf)

2

Agent Architecture

Lernziele

- Agentenarten: reaktive, proaktive, ...
- Umsetzung von Agentenarchitekturen
  - Table-Based
  - Reflexive/Proactive
  - Model-/Goal-/Utility-based
- Problem-solving vs. Planning
- Agent States

2.1 Agent Architectures

Literature distinguishes a wide scale of agent types. Providing an exhaustive list here is neither informative nor feasible. Please refer to [**wooldridge1995intelligent**] for a more complete classification of agents.

For we start with the following list.

Agent Architecture Summary

- Table-driven agent
- Reflex agent
- Model-based agent
- Goal-based agent

- Utility-based agent

2.1.1 Table-Driven Agent

Create a table mapping all possible percept sequences onto actions and you have a table-driven agent. It obviously is not feasible to write down all — possibly infinite — percept sequences and thus this type of architecture has to stay, except for very limited problems, a theory

2.1.2 Reflexive Agent

A reflexive agent is "reacting" to sensor measurements. [**russell2010artificial**]

A reflex agent is a stateless reactive machine that evaluates a current percept and selects an action based on this percept.

Simple Reflex Agent

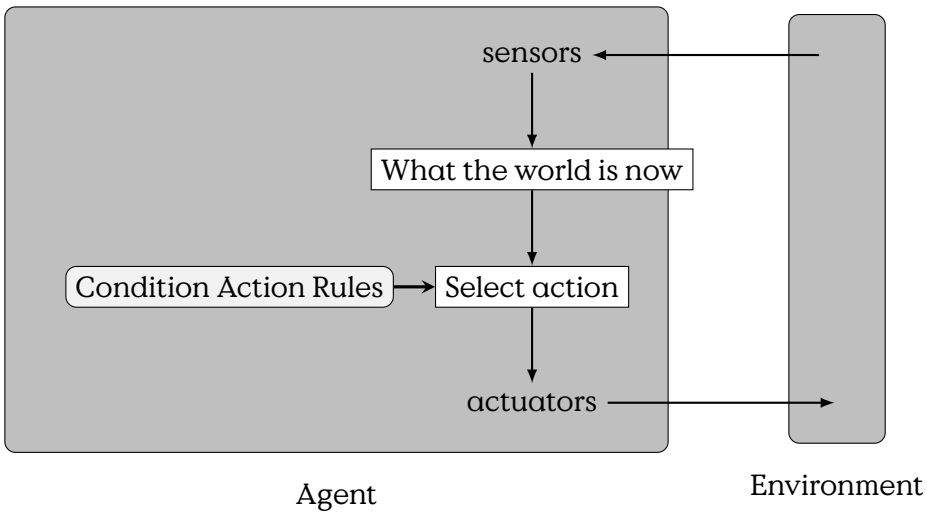


Figure 2.1: Simple Reflex Agent Model

function REFLEX-AGENT(percept)
persistent:
rules: condition-action-rules
state←INTERPRET-INPUT(percept)
rule←RULE-MATCH(state, rules)
action←rule.ACTION **return** action
end function

Proactive Agent

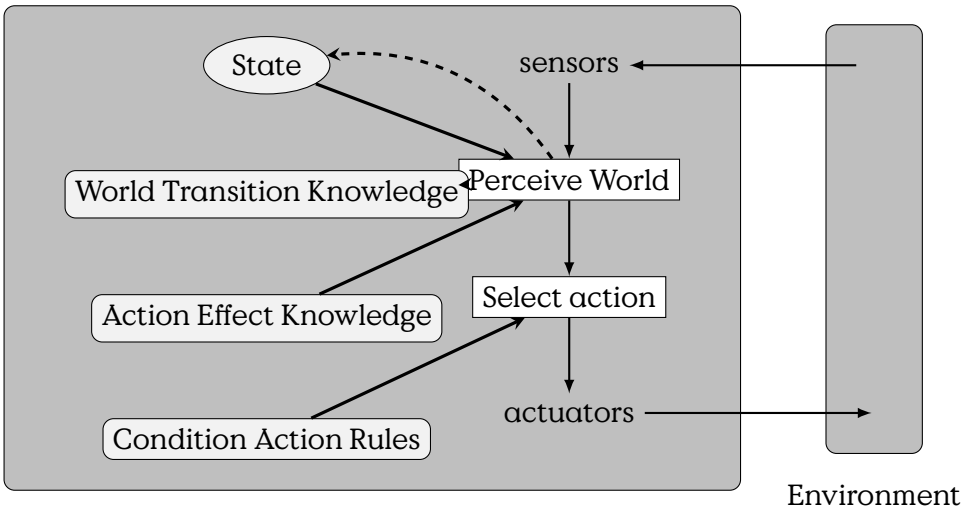
A proactive Agent differs from a reflexive agent insofar as it is a type of agent that not only is "waiting" for a the environment providing a percept, but can proactively act, for example guided by an internal clock. We could argue that a clock-measurement can be modelled as sensor input and proactivity does not necessitate a distinction.

2.1.3 Model-driven Reflex Agent

The difference between a reflex agent and a model-based reflex agent is that the latter includes an internal state. This state can be used to store a model of the world, derived and extended by a percept sequence. Generally every stored perception already can be seen as part of a world model.

More on states is found in Section **??**.

Model-based Reflex Agent



Model-Based Reflex Agents

function MODEL-BASED-REFLEX-AGENT(percept)
persistent:
state: current world-model,

transition_model: description of world-transition based on current state and action,
sensor_model: description of state-transition based on percepts,
rules: condition-action-rules

state←UPDATE-STATE(state, percept, transition_model, sensor_model)
rule←RULE-MATCH(state, rules)
action←rule.ACTION
return action
end function

2.1.4 Goal-based Agent

A goal-based agent is not only selecting actions based on percepts, but includes an filter, that can either exclude actions that are not in line with its goals. In order to do this, the agent must be able to predict the outcome of its action, i. e., it must calculate the outcome based on its knowledge on the effect of actions on the world and its perception of the world. This is modelled as an additional decision step within the agent.

Goal-based Agent

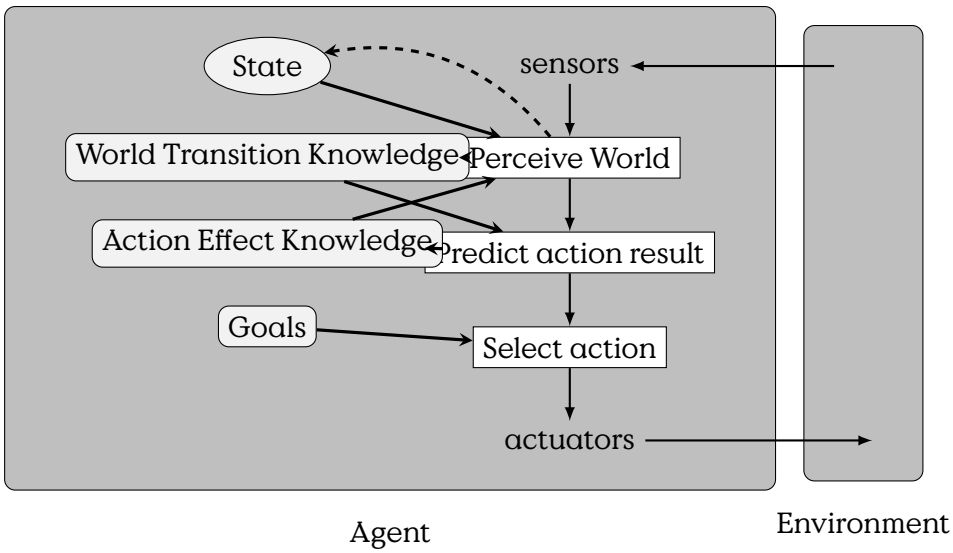


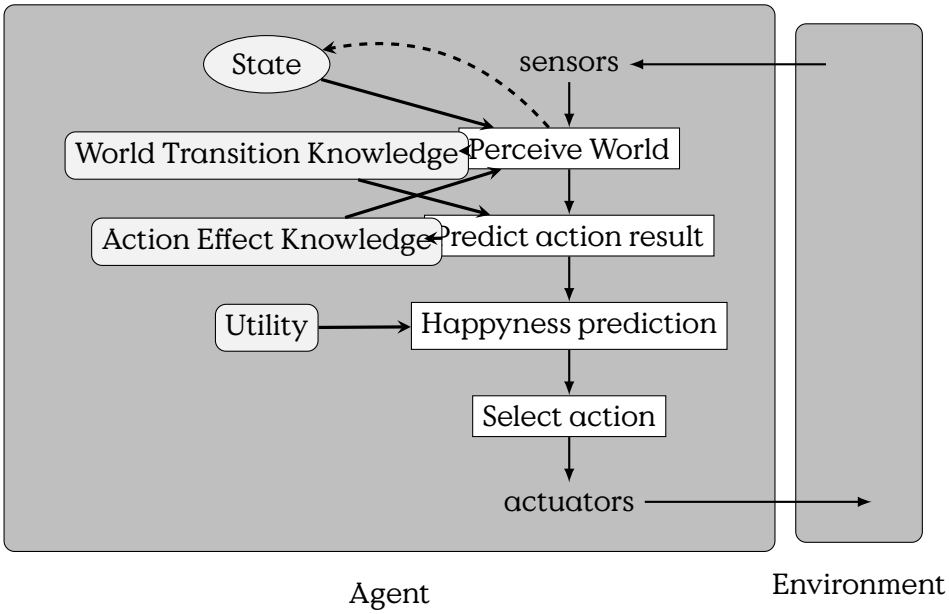
Figure 2.2: Model-based, goal-based agent

2.1.5 Utility-based Agent

But goal-based agents can only decide on whether an action would achieve a goal or not. This binary decision provides sufficient information to determine whether an action would not be in line, essentially finding undesirable action, but it cannot decide between desireable actions. To achieve this, an agent requires an explicite measure of happyness.

The term utility is a model for value, e. g., usefulness of a tool, or risk-level of a given activity. For quantification utility is implemented as an utility function.

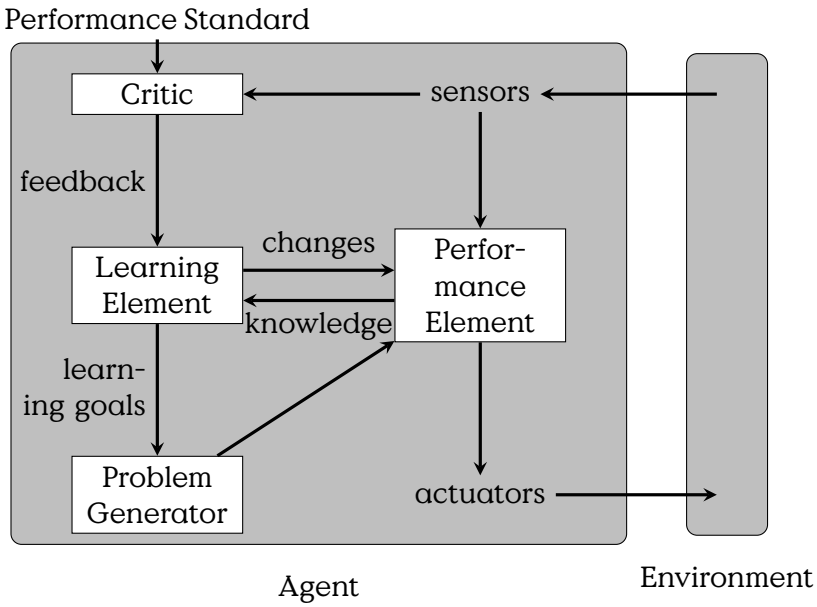
Utility-based Agent



2.1.6 Learning Agent

A Learning is an orthogonal category to the previous agent categories. A learning agent is able to derive new rules from the perceived world, guided by an external performance standard.

Learning Agent



performance element selection of actions, might be implemented as one of the "normal" agent models

learning element responsible for adoption or evolution of rules

problem generator suggests actions in order to explore new experiences

critic evaluation of performance and determines changes to the performance element

2.2 Agent State

The expressiveness of the state of an agent determines what an agent can remember about the world. For some problems it is sufficient to be able to distiguish a finite set of states, e. g., navigation between cities. But more complex may require that states can be attributed with quantities, e. g., power remaining, or even a representation of relationships between perceived objects.

2.2.1 Atomic State

2.2.2 Factored State

2.2.3 Structured State

2.3 CASiMiR

How can we decide how whether a problem is suitable for the agent model? Well, probably some experience would with that, but we could find some more distinguished arguments.

CASiMiR

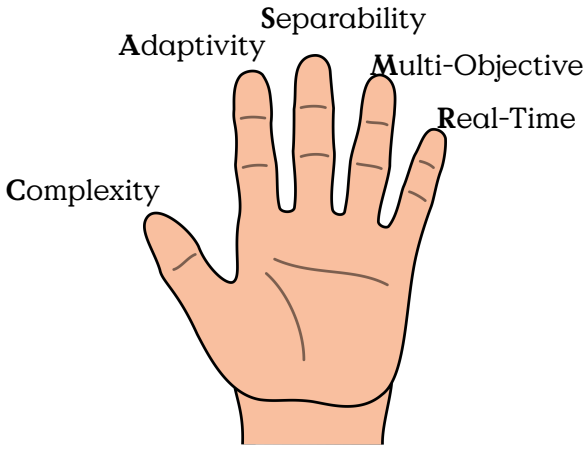


Figure 2.3: Categories for structured decisions on the suitability for an agent-based solution of a given problem. [Hand Open Heavy Outline by Schplook, openclipart]

2.4 Summary

3

Constraint Satisfaction Problems

Lernziele

- Problem als Constraint Satisfaction Problem (CSP) definieren können
- Constraint-(Hyper-)Graph erstellen
- Consistency-Arten kennen
- CSP Löser
  - AC-4
  - Backtracking/Distributed Backtracking

Searching vs. Constraints

- Search algorithms
  - Breitensuche
  - Tiefensuche
- State-Space Search
  - Nodes are states
- Algorithm:
  1. Goal reached?
     - Yes: STOP

  - No: select next
- Define state-constraints
- Oftentimes easier to
  - define (locallity)
  - fast (pre-)evaluation

  e. g., Map-Colouring

CSP

Definition 3.0.1 (CSP)**.**

$$\begin{aligned}\mathcal{X} &: \{X_1, \ldots, X_n\} \\ \mathcal{D} &: \{D_1, \ldots, D_n\} \\ \mathcal{C} &: \{\langle scope, relation\rangle\}\end{aligned}$$

$\mathcal{C}$-Notation: $\langle (X_i, \ldots, X_j), f : D_i \times \cdots \times D_j \rightarrow bool\rangle$

Example: Graph Colouring

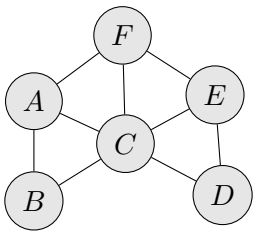


Figure 3.1: Example Graph Colouring Problem

$$\begin{aligned}n =& \{A, B, C, D, E, F\} \\ e =& \{C_{AB}, C_{AC}, C_{AF}, \\ & C_{BC}, C_{CE}, C_{CD}, \\ & C_{CF}, C_{DE}, C_{EF}\}\end{aligned}$$

$$\forall C_{xy} \in e : C_{xy} = \langle (x, y), x \neq y\rangle$$

A domain D_i of a variable X_i is the set of valid values. During evaluation domains are revised, i. e., values that do not satisfy a constraint are removed from the domain. Iteratively domains are reduced until — ideally — a domain contains only one acceptable value. If a domain is reduced to zero elements then there is no solution for the given Constraint Satisfaction Problem (CSP).

Domains

Global constraints usually concern all, or at least a large number of variables.

Global Constraints

Equality $Alldiff(X_i, \ldots) := \forall i, j : i \neq j \Rightarrow X_i \neq X_j$

Resource $Atmost(X_i, \ldots) := \sum_{X_i} \leqslant threshold$

Bounds-Propagation

Variables bound by:

$$\langle (X_1, X_2), X_1 + X_2 = k \rangle$$

Domains:

$$D_1 = [l_1, h_1]$$
$$D_2 = [l_2, h_2]$$

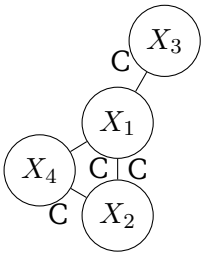
then revise domains as

$$D_1 \leftarrow [k - h_2, h_1]$$
$$D_2 \leftarrow [k - h_1, h_2]$$

3.1 Constraint Graphs

Constraint Graph

- Variables X_i as nodes
- Constraints C_j as edges



Constraint Hyper-Graph

Problem:

- Letters represent digits
- Words represent numbers
- Find substitution Letter-to-Digits
- Each digit only once

	T	W	O
+	T	W	O
F	O	U	R

Constraints?

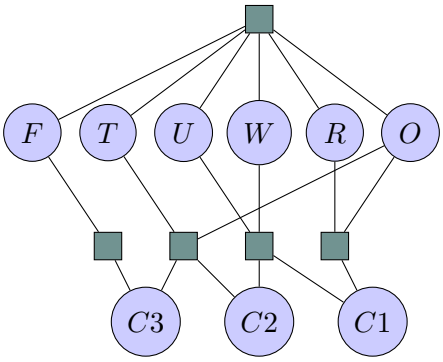


Figure 3.2: Hypergraph of cryptarithmetic puzzle TWO-TWO-FOUR.

CSP are much easier to solve if all constraints are (at most) binary relations. We can turn any constraint graph into a binary constraint graph by using the graph dual.

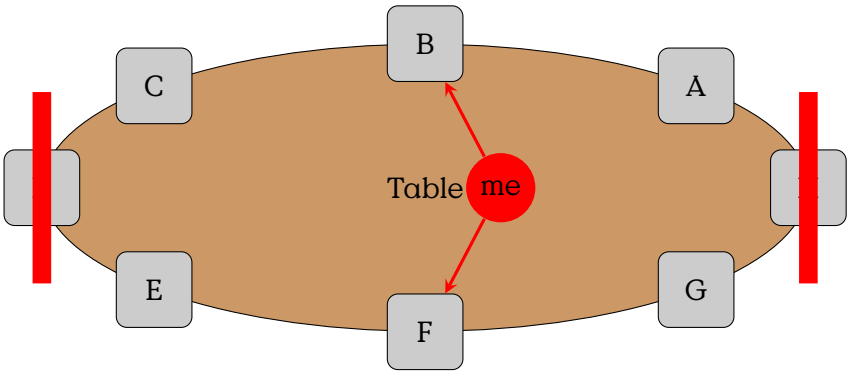
Each edge is turned into a variable and an edge is created for every variable connected to that edge.

Binary Graph Construction

1. $\mathcal{C} \rightarrow N$
2. $\forall C \in \mathcal{C} : \forall X \in scope(C) : e_{C,X} \in E$

In this example you have invited a set of friends. You have eight seats at your table, but then the situation escalates. Can you formulate constraints?

Example: Seating Friends



1. Friends: Tom, Finn, Lisa, Ida, Max, Elsa, Grace
2. I sit at B or F
3. Tom and Max cannot sit on adjacent seats
4. Doors block D and H

3.2 Consistency

Different type of consistencies by the locality of the constraints fulfilled. Locality is determined by the scope of a constraint.

- Node consistency
- Arc Consistency
- Path Consistency
- K-Consistency
- Global Consistencies
  - $Alldiff$
  - $Atmost$

Consistencies

Node single variable $\langle (X), \ldots \rangle$

Arc edge between nodes $\langle (X, Y), \ldots \rangle$

Path X, Y connected by Z $\langle (X, Z), \ldots \rangle, \langle (Z, Y), \ldots \rangle$

K-consistency Path of constraints with n elements

- $i \in \{1, \ldots, n\} : \langle (X_i, X_{i+1}), \ldots \rangle$

3.3 CSP-Solving

Solution algorithm to a CSP do not necessarily produce a unique solution to a problem but is used to efficiently reduce the state-space that has to be searched for a valid solution. After reduction of states, a suitable state-space search-algorithm is used to find one or all solutions, depending on the application, that satisfy the given constraints.

Solving CSP

Consistence Revision
- Reduction of state-space
- Reduce domains of variables
- Often: first reduction is easy

State Search
- Various Methods
  - Depth/Breadth-first
  - Best-first
  - Heuristic search
    - e. g., A*

Solving of arc-constraints can be done using the algorithm AC-3. The algorithm starts with a queue containing all arcs of a CSP — each direction individually. It then successively removes values from the domain of the first node in each arc. If during this step a domain is changed, i. e., "revised", then all outgoing arcs of the related node are pushed onto the queue. (See Figure **??**)

AC-3

3.4 Distributed CSP

Distributed CSP

- Agents communicate
  - Input used for domain pruning
- Simple Solution (Domain pruning)

function AC-3(csp) **return** false if inconsistency found, true otherwise
  $queue \leftarrow$ all arcs in csp
  while $queue$ not empty **do**
    $(X_i, X_j) \leftarrow POP(queue)$
    if $REVISE(csp, X_i, X_j)$ **then**
      if sizeof(D_i) $= 0$ **then return** false
      end if
      for all $X_k \in X_i.neighbours()/\{X_j\}$ **do**
        $add(X_k, X_i)$ to $queue$
      end for
    end if
  end while
return true
end function
function REVISE(csp, X_i, X_j) **return** true iff D_i changed (revised)
  $revised \leftarrow false$
  for all $x \in D_i$ **do**
    if no value $y \in D_j$ with (x, y) satisfies constraint $X_i \leftrightarrow X_j$ **then**
      delete x from D_i
      $revised \leftarrow true$
    end if
  end for
end function

Figure 3.3: AC-3 Domain Pruning Algorithms for Path-Consistency (see [**russell2010artificial**])

- Communicate whole domain
- Filter own domain: remove conflicts
- Domain contains single value $\Longrightarrow$ solution found

relatively "weak" approach

Backtracking

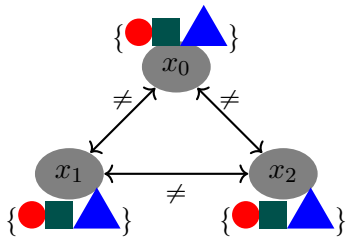
function BACKTRACK($csp, assignment$) **return** solution or failure
  if complete($assignment$) **then**
    add $\{var = value\}$ to $assignment$
    $inferences \leftarrow INFERENCE(csp, var, assignment)$
    if $inferences\neg$failure **then**
      add $inferences$ to csp
      $result \leftarrow BACKTRACK(csp, assignment)$
      if $result\neg$failure **then return** $result$
      end if
      remove $inferences$ from csp
    end if
    remove $\{var = value\}$ from $assignment$
  end ifreturn failure
end function

Figure 3.4: Backtracking algorithm [**russell2010artificial**]

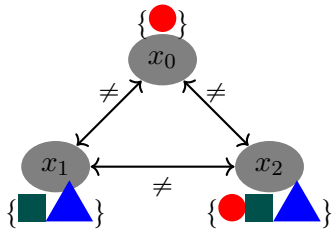
Backtracking in Dist-CSP

function DIST-BACKTRACK
  select $val \in domain$
  repeat
    if $\neg consistent(val, neigh(val)$ **then**
      remove val from $domain$
      select $val \in domain$
      notify neighbours
    else $\triangleright$ consistent val or no consistency possible
   return do nothing
    end if
  until val has not been changed
end function

Figure 3.5: Distributed Backtracking algorithm [**shoham2010multiagent**]



Node x_0 chooses



Problems of Dist-CSP

- Agents decide locally
- No perfect knowledge about others
- Problems:

  Snapshot-Problem How did others decide?

  Termination Are we done yet?

  Convergence Is the global solution valid?

Application CSP

- CSPs provide approach to problems, e. g.,
  - creating a timetable
  - assignment of teachers to topics and classes
  - machine scheduling
- often CSPs are not sufficient
  - existing soft-constraints
  - weighted constraints
  - "valid" is not sufficient, "optimal" is needed.

4

Optimization

4.1 Optimization Problems

Based on Lecture "Agentenbasierte Verfahren in Energiesystemen", Astrid Niesse, Winter Term 22/23.

Optimization Problems

- Typical engineering problems have a process that can be represented by a mathematical model.
- A performance criterion is given (e. g., minimum cost, maximum efficiency).
- Goal of optimization is to find the values of the process variables that yield the best value of theperformance criterion.

Typical performance criteria

- maximum profit
- minimum cost
- maximum efficiency effort
- minimum error
- maximum throughput
- best product quality

Optimization Strategies

Static optimization variables have numerical values, invariant with respect to time

Dynamic optimization variables are functions of time

Elements of optimization problems:

1. (At least) one optimization function $f(vars)$
2. Equality constraints $h(vars) = s$
3. Inequality constraints $h(vars) \leqslant t$

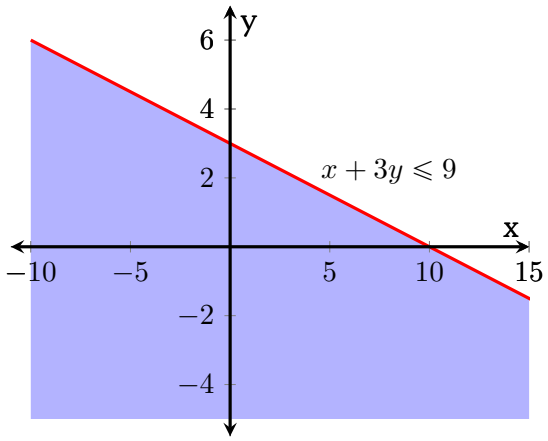


Figure 4.1: Example: Two-dimensional inequality constraint.

Optimal Solutions

- An optimal solution is an assignment to process variables that is within the feasible region and provides the best value of the performance criterion.
- A meaningful optimization problem usually is underdetermined, i. e., has at least one degree of freedom.
  - Optimize (maximize/minimize) $f(x)$
  - Subject to $\begin{cases} h(x) = 0 \\ g(x) \leqslant 0 \end{cases}$
  - with x vector of n variables $(x_1, x_2, \ldots, x_n)$,
    - $h(x)$ vector of equalities
    - $g(x)$ vector of inequalities.

Solving Optimization Problems

1. Analyse the process and make a list of all variables
2. Determine the optimization criterion and specify objective functions
3. Develop the mathematical model of the process
   (a) define the equality and inequality constraints
   (b) Identify dependent and independent variables (degrees of freedom)
4. If the problem formulation is too large or complex simplify it if possible (not detailed here!)
5. Apply a suitable optimization technique (which ones?? Later...)
6. Check the result and examine it's sensitivity to changes in model parameters and assumptions

Kurze Wiederholung Statistik. Der Begriff sensitivity (Sensitivität)

Sensitivity and Specifity

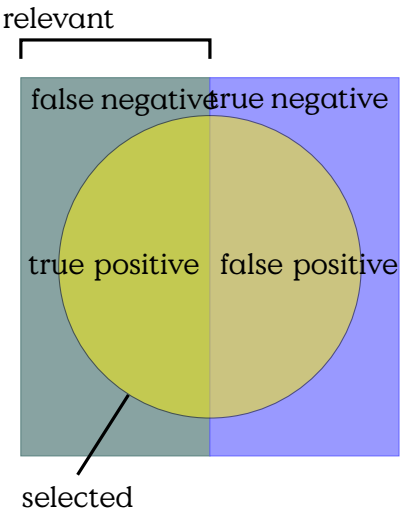


Figure 4.2: Statistical testing, Sensitivity and Specifity

Sensitivity $TPR = \frac{TP}{TP+FN}$

Specifity $TNR = \frac{TN}{TN+FP}$

Precision $PPV = \frac{TP}{TP+FP}$

Accuracy $ACC = \frac{TP+TN}{P+N}$

Notation

P, N Positive/Negative Outcomes

TP, TN Correctly classified (True) positive and negative outcomes

FP Falsely as relevant (positive) classified negative outcomes.

FN Falsely undetected relevant (negative) classified relevant (positive) outcomes.

TPR True Positive Rate: Ratio of elements classified as relevant to all existing relevant elements. Usually cannot be determined except in controlled experiments where the actual set of relevant elements is known beforehand.

TNR True Negative Rate: same as for TPR except for negative results.

PPV Positive Predictive Rate: relative number of true positives among all elements classified as positive. Precision generally measures how narrow the results of a classifier are. That is not the same has having correct (or accurate) results.

ACC Accuracy: relative number of correctly classified outcomes of a test with respect to total number of outcomes — be it positive or negative.

<++>

4.2 Hill-Climbing

[russell2010artificial]

Hill climbing is a basic algorithm to get from an initial state, e. g., a random state, towards the maximum of the current local slope. In a more complicated landscape this will not always be the global maximum. It can even be, that it is a very minor maximum.

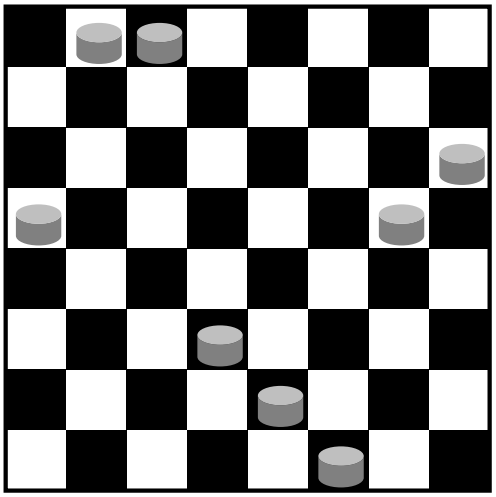
Hill-Climbing

Greedy Local Search

function Hill-Climbing(problem) **return** state that is local maximum
  $cur \leftarrow state.INITIAL$
  while true **do**
    $neigh \leftarrow$ highest-valued successor state
    if $val(neigh) \leqslant val(cur)$ **then return** cur

end if
$cur \leftarrow neigh$
end while
end function

Example: Hill-Climbing



Problems of Greedy Searches

Local Maxima

Ridges

Plateaus Shoulders and Flat Local Maximum

Optimization: State Space Landscape

4.3 Simulated Annealing

[russell2010artificial]

Die Idee hinter Simulated Annealing ist, den Zustandsraum immer wieder zu "schütteln" und so, dass der aktuelle Zustand auch mal in Richtung schlechterer Lösung verschoben werden kann. Mit der Anzahl der Iterationen wird ein solches Verschieben zum Schlechteren immer unwahrscheinlicher und der Algorithmus wird dem Hill-Climbing (Abschnitt **??**) immer ähnlicher.

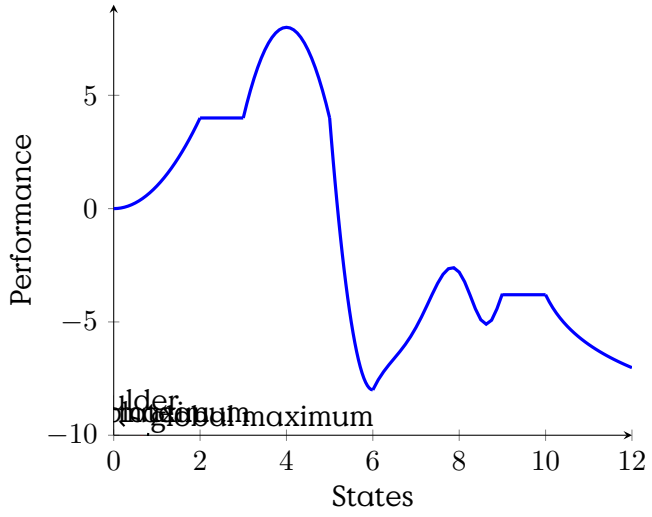


Figure 4.3: One-dimensional state-space landscape. Elevation represents the value of the objective function/performance. (See [**russell2010artificial**])

Eine kleine Eigenart ist noch, dass die Richtung des Optimum umgedreht wird und das globale Minimum gesucht wird.

Simulated Annealing

function Simulated-Annealing(problem, schedule) **return** solution state
  $cur \leftarrow problem.INITIAL$
  for $t = 1$ to ∞ **do**
    $T \leftarrow schedule$
    if $T = 0$ **then return** cur
    end if
    $next \leftarrow$ random successor of cur
    $\Delta E \leftarrow val(cur) - val(next)$
    if $\Delta E > 0$ **then**
      $cur \leftarrow next$
    else
      $cur \leftarrow next$ with probability $e^{\Delta E/T}$
    end if
  end for
end function

4.4 Local Beam Search

Anstatt nur einen einzelnen Zustand zu verfolgen, werden beim local beam search k Zustände parallel optimiert. Dabei wird die Optimierung nicht einfach parallelisiert, sondern es werden jeweils immer die k besten Nachfolger aller aktuellen k Zustände als nächster Zustand ausgewählt. Dadurch wird, auf Kosten der Diversität, sehr schnell schlechte Lösungen verworfen. Dieses Problem lässt sich dadurch abmildern, indem die k nächsten Zustände nach gewichtetem Zufall ausgewählt werden.

5

Distributed Optimization

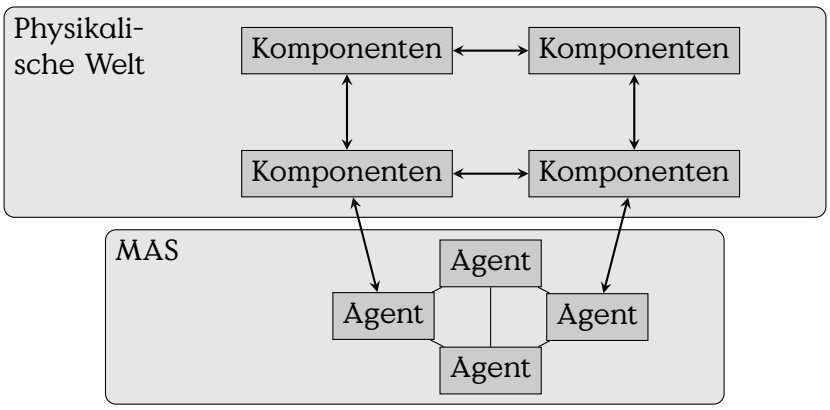
AntNet

- Self-Organization using indirect agent communication
- AntNet probabilistic routes

Multiagentensysteme sind verteilte Systeme in dem Sinne, dass die Lösung eines (oder mehrerer) Probleme auf unabhängige Ausführungseinheiten aufgeteilt wird. Dabei übernehmen unterschiedliche Agenten unterschiedliche Teilaufgaben deren Ergebnisse durch Kommunikation zusammengeführt werden.

> a multiagent dynamic system can be defined as a network of a number of loosely coupled dynamic units that are called agents [**shoham2010multiagent**]

Multi-Agent System (MAS)



Agentensysteme: Designentscheidung

1. Directory services
    - How to find other agents? Yellow-page Agent/Environment
2. Communication infrastructure/topology
    - How to interact with other agents?
    - Blackboard(bus), point-2-point, multi-cast,...
3. Protocols and common understanding
    - Format of messages?
    - What is the meaning? Semantics!
4. Security issues
    - Authentication and message integrity

Network Overlay

- **Overlay-Topology**: Kommunikationsschicht oberhalb anderer Kommunikationsschichten
    - ...beschreibt direkte Verbindungen auf dieser Abstraktionsschicht
    - Kapselung von
- Beispiel: Internet über Ethernet
    - “eigentliche” P2P-Kommunikation: Ethernet
    - Internet ist Overlay, überbrückt Netzsegmente
    - darüber dann wiederum DNS-basierte Adressen
- Umsetzung/Struktur
    - Orientierung an Kommunikationsstruktur
    - ...anhand physischer Welt
    - ...unabhängig davon.
    - statisch vs. dynamisch

Die Klassifizierung von Netztopologien basiert auf unterschiedlichen Kategorien von Eigenschaften. Graphen, das gängige Modell zur Abstraktion von Verbindungsstrukturen, werden klassifiziert anhand ihrer Zentralität bzw. Verteiltheit, dem Durchmesser, der Verteilung der Anzahl

der Kanten an den Knoten (Degree), der Verbundenheit, und anderen Eigenschaften.

Network Topologies

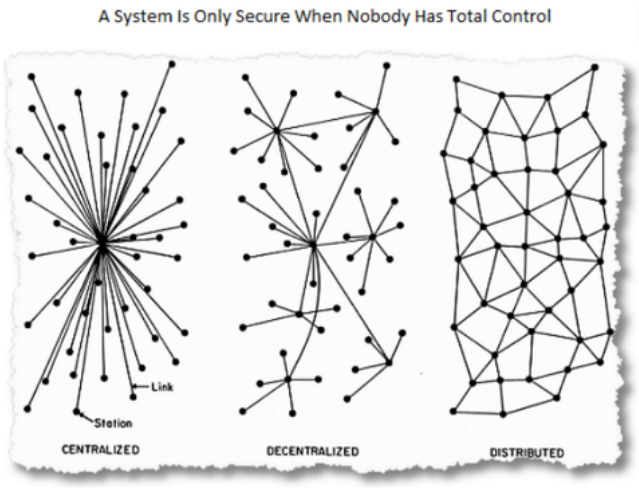


Abbildung 5.1: Topolgies in different degrees of distributedness

5.1 AntNet

This lecture introduces AntNet algorithms inspired by self-organization of ant-hives. Main source for this part is Muddassar Farooq and Gianni A. Di Caro: "Routing Protocols for Next-Generation Networks Inspired by Collective Behaviors of Insect Societies: An Overview".[**FarooqDiCaro2008**]

Problem

- Shortest Path without global information.
- e. g., Mobile Wireless Mesh Network
- Decision at node s
  - Which node $1, 2, 3$ is closest to d?
  - No information on Network available!
- How to determine route without knowing the graph?

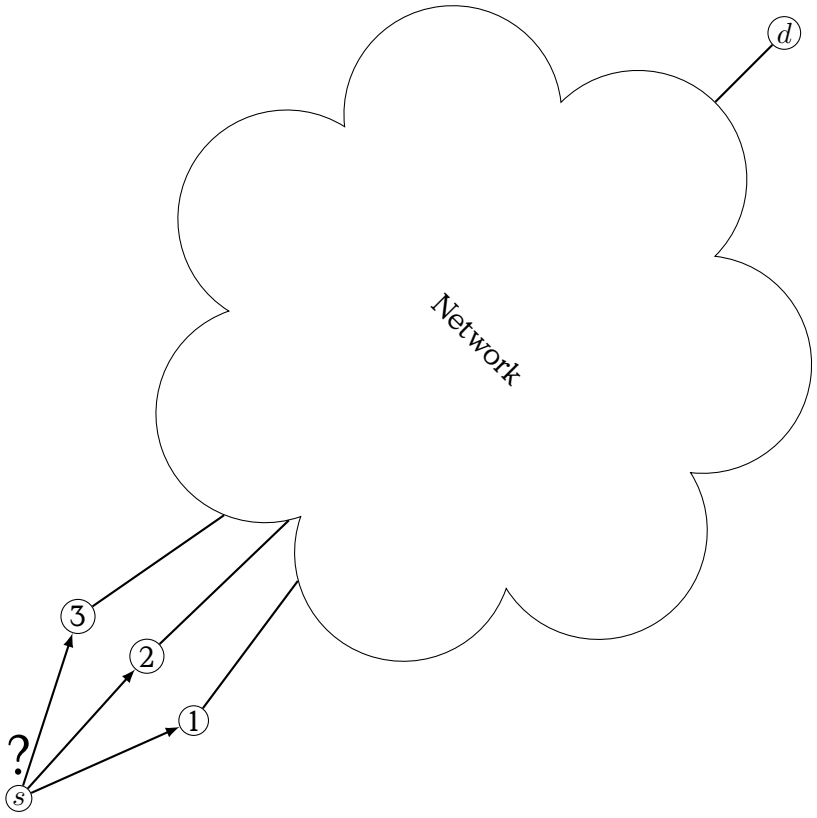


Figure 5.2: Routing Problem

Biological Antetype

The fundamental idea of AntNet is to log the results of proactive path exploration, using mobile agents called "ants". Ants are small packages that use a specific routing heuristic to decide on which node to jump to next.

The forward ant is propagated from source s to destination d choosing the next hop node randomly from the list of unvisited neighbours of the current node. It logs the visited nodes in sequence, and the travel-time of individual hops.

AntNet

node router in the network

neighbour node directly reachable from current node

ant specific packet/agent exploring the network of nodes

forward ant probabilistic exploration to destination d

backward ant update nodes on source route to s

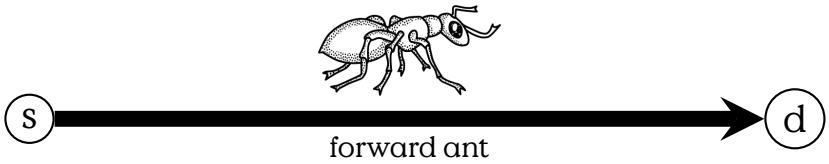


Figure 5.3: AntNet Forward Ant

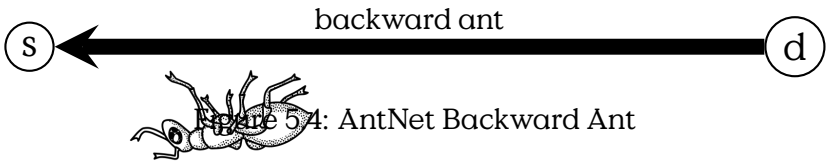


Figure 5.4: AntNet Backward Ant

[Image "Ant" by frankes https://openclipart.org/detail/214038/ant-lineart]

Routing Node i

Statistical Model(World Model)$\mathcal{M}^i$: $N \rightarrow (\mu_d, \sigma_d, W_d)$

Pheromone Table$T^i\mathcal{N}^i \times N \rightarrow [0,1]$

Routing Table$\mathcal{R}^i$: $\mathcal{N}^i \times N \rightarrow [0,1]$

Link Queue Length$\mathcal{L}^i$: $\mathcal{N}^i \rightarrow \mathbb{N}$

Nodes...

- regularily generate ants for exploration to arbitrary destinations.
- route packets using $\mathcal{R}$ and
- route ants using $\mathcal{T}$.
- Backward ants update $\mathcal{T}$.

Every node in our network keeps a routing matrix $\mathcal{R}$ (not "routing table"), that provides a quality estimate for each neighbour node n and every potential destination node d. For each destination, the set of matrix entries over all neighbours resembles a probability measure, i. e., $\sum_{n \in \mathcal{N}^i} \mathcal{R}^i(n, d) = 1$ for a given d.

The router further has knowledge of the length of the outbound queues $\mathcal{L}^i$ towards the different neighbours. This information is further used to estimate the utility of a neighbour as next hop towards a given destination.

In other words, when a packet arrives at a routing node, the router selects a neigbouring node, choosing with higher likelyhood a neighbour that provides a better quality value for the given destination.

A node further keeps a statistical world model $\mathcal{M}^i$ of mean μ, standard

deviation σ and best time W to every node. We will see how this is needed later, when we discuss the pheromone update by the backward ants. And finally there is the pheromone matrix $\mathcal{P}^i$ which is a specific routing table used for routing forward ants. The pheromone matrix will be updated by backward ants and the normal routing matrix is derived from the pheromone matrix.

Exploration Algorithm

Ant 
source s destination d path $\mathcal{P}$ link_quality $T_{i \to j}$

Figure 5.5: AntNet Ant Class

- Hop selection propability at Router

$$p_{nd} = \frac{t_{nd} + \alpha l_n}{1 + \alpha(|\mathcal{N}^i| - 1)}$$

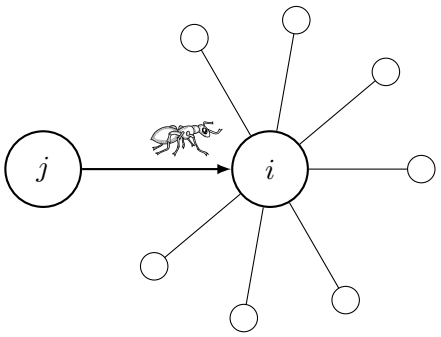


Figure 5.6: AntNet Exploration, ant arriving at node i coming from node j

Ant arrives at node i from node j

1. log i and transfer time $t_{j \to i}$
2. read π_{nd} for all neighbours $n \in \mathcal{N}^i$
3. make weighted random choice for next hop
4. transfer to next hop

An ant in forward mode is very straight forward. It randomly chooses a next hop, avoiding nodes already visited, and repeats until it reaches its destination. The interesting part happens when it reaches d. At that point it switches into backward mode and backtracks its path from its memory, updating the local pheromone matrix at every hop.

Update Algorithm

1. follow the path backwards
2. at every hop update T^i

Arriving from node j a backward ant is changing the pheromone table. For the final destination d and every intermediary on the path from i to d, denoted δ, the ant is reinforcing this path and weakening all other paths.

Reinforce path over neighbour that backward ant came from:

$$\begin{aligned} t_{j\delta} &\leftarrow t_{j\delta} + r(1 - t_{j\delta}) \\ t_{k\delta} &\leftarrow t_{k\delta} - r t_{k\delta}, \forall k \in \mathcal{N}^i, k \neq j \end{aligned} \tag{5.1}$$

Reinforcement value r is a weighted sum of the ratio between the best time and the measured time and the XXX-function F.

$$r = c_1 \frac{W_\delta}{T_{i\delta}} + c_2 F(T_{i\delta}, \mu_\delta, \sigma_\delta^2, W_\delta) \tag{5.2}$$

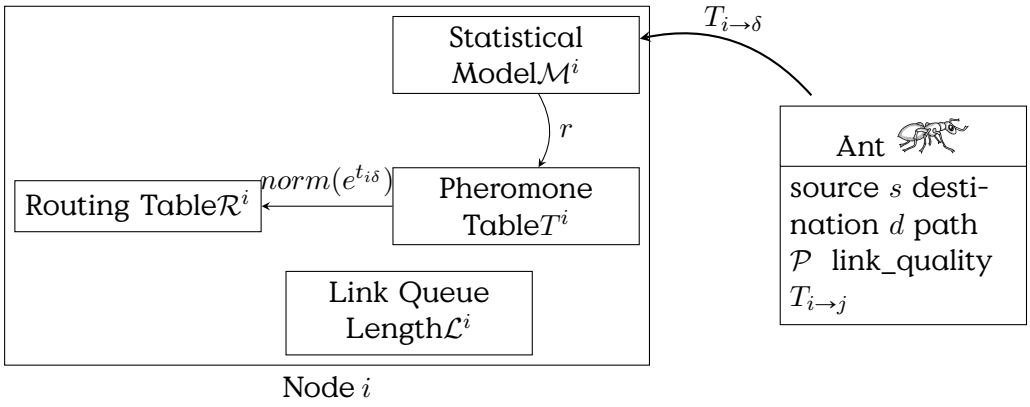


Figure 5.7: Updating the pheromone table at an node

l is an instantanious estimate of the quality of a next hop, providing information about the current congestion situation. While the pheromone matrix provides a long-term estimate.

Agent Model

What kind of agent is the ant?

- Reflexive?
- Proactive?
- Rational?
- Deterministic?

Example

Router Class Diagram

AntRouter
Pheromone Table T^i Statistical Model $\mathcal{M}^i$

PaketRouter
Routing Table $\mathcal{R}^i$

$$\sum_{n \in \mathcal{N}^i} \mathcal{R}^i_{nd} = 1 \tag{5.3}$$

AntNet nodes consist of two related routers. The routing tables $T^i, \mathcal{R}^i$: $\mathcal{N}^i \times N \rightarrow [0, 1]$ define a probability for every neighbour being selected as next hop given the final destination of a packet. T^i used for routing forward ants and modified by backward ants. $\mathcal{N}^i$ is used for routing packets. $\mathcal{N}^i$ is derived from T^i by appliing the exponential function individually onto each entry and then normalising the values for each destination d so that the sum is equal to one, see Eq. **??**.

Notation
$\mathcal{R}$ Routing Table
$\mathcal{N}^i$ Neighbours of node i

5.2 Particle Swarm Optimization (PSO)

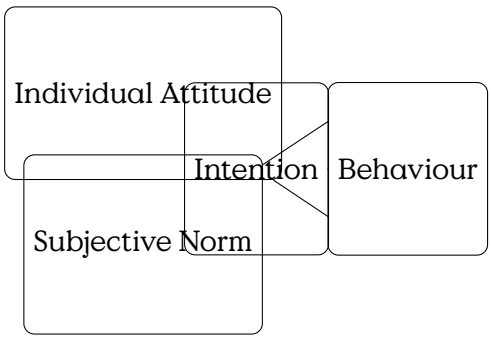
Der Begriff PSO umfasst verteilte Algorithmen bei denen unabhängige Einheiten, also Agenten, sich durch wechselseitige Beobachtung koordinieren. Wenn dies klappt, dann agieren diese Agenten als Schwarm, i. e., sie verhalten sich als große Einheit.

Die biologischen Vorbilder sind das Bewegungsverhalten in Fisch- oder Vogelschwärmen. Durch wechselseitige Beobachtung und Anpassung von Geschwindigkeit und Richtung der eigenen Bewegung an die der Nachbarn wird erreicht, dass der gesamte Schwarm sich als Einheit bewegt.

Die Grundidee ist, dass sich das Verhalten einer Einheit aus seiner Intention (Ziele und Einschränkungen) bestimmt. Die Intention wiederum wird durch eigene (intrinsische) Motivation und die Ziele der sozialen Gruppe bestimmt.

Particle Swarm Optimization (PSO)

Reasoned Action Model



[**kennedy01swarmintelligence**]

- Behavious as function of Intention
- Subjective Norm: Social rules
- Individual Attitude (towards behaviour)

Die Definition des Partikel Schwarms findet sich in [**kennedy01swarmintelligence**]. Dabei optimiert jedes Particle (=jeder Agent) seine eigene Position durch Anpassung des eigenen Geschwindigkeitsvektors und damit seiner nächsten Position.

Particle Swarm

$$v_{i+1} = \omega v_i + \varphi \otimes (p - x_i) \tag{5.4}$$
$$x_{i+1} = x_i + v_{i+1} \tag{5.5}$$

where

- x, p positions in system space
- p is "best point", i. e., intention
- φ is weight towards p
- v is rate of change of agent

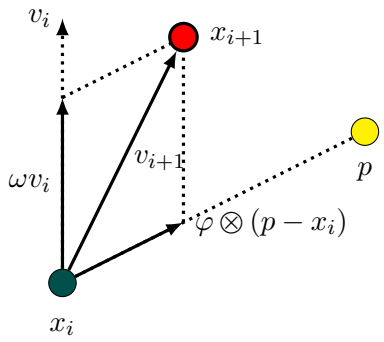


Abbildung 5.8: PSO adopting v towards the direction of $p - x_i$

Gleichung **??** zeigt ein grundsätzliches Modell für die schrittweise Anpassung der aktuellen Geschwindigkeit an.

Deconstructing Intention

Distinguishing individual and group objectives

$$p = \frac{\varphi_i p_i + \varphi_2 p_g}{\varphi_1 \varphi_2} \tag{5.6}$$

Geographic Routing

Definition 5.2.1. Finding a route in a network, where node addresses are values in a geometric space, i. e., a vector space

[travel map by vectorsme, 2013-07-28]

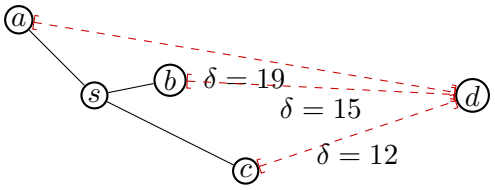
- "Internet-Space": Hierarchical
- Geometric Space: normed (denoted positions)
- Nodes $\mathcal{N}$ know own position and neighbours' position
- Source s knows destination position d
- Packets contain $O(1)$-limited historic transit data
- Also: "Position-based routing"

Georouting Applications

- Wireless Mobile Ad-hoc networking
- Sensor Networks
- Vehicular Networks
- Related: Opportunistic Routing (Mobile Networks)

Greedy Routing

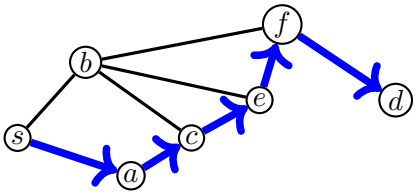
- Positions from metric space
- Route from source s
- Next hop = closest to destination d
- Known:
  - Current node s, without saying that there is not a real source where a packet originally originated, but for simplicity reasons we use the common notation for source for the current node.
  - Neighbours $N_s \subset \mathcal{N}$
  - Neighbours' positions
  - Destination d position
- Calculate distances $\delta : N \to \mathbb{R}$



Greedy Routing Algorithm

Current node: n_i, Destination d

1. Calculate $d(n, d)$ for all $n \in N_{n_i}$
2. Choose n_{i+1} with minimum $d(n_{i+1}, d)$
3. Forward packet to n_{i+1}



Distance Metrics

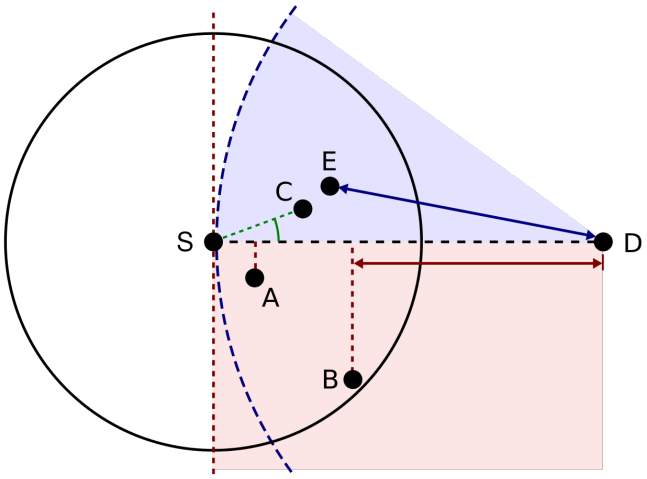


Figure 5.9: CC BY-SA 3.0, Greedy forwarding variants by Stefan Ruehrup @wikipedia

A Nearest Forwarding Progress (NFP), reduces energy-requirements for communication in wireless networks

B Most Forwarding Progress (MFP), reduces (hopefully) number of communication hops, stays close to optimum line

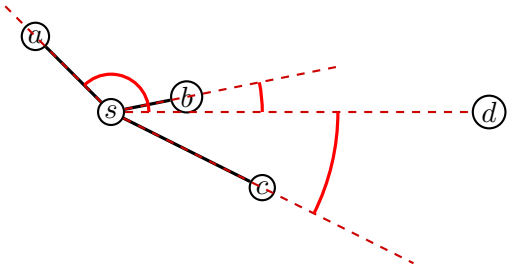
C Closest Angle, i. e., Compass-Routing

E Greedy, Neighbour with shortest distance to destination

<++>

Compass Routing

- Route to neighbour $\in N_s$ with lowest angle to destination
- Compare, choose lowest
  - $\vec{sc} \angle \vec{sd}$
  - $\vec{sb} \angle \vec{sd}$
  - $\vec{sa} \angle \vec{sd}$
- Not loop-free

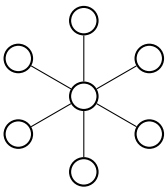


[Compass rose 2 by Firkin 2015-11-29, 11:23 pm]

Greedy Routable Embedding

- Positioning of nodes $\mathcal{N}$ in a way that
  - $\forall(s,d) \in \mathcal{N} \times \mathcal{N}$
  - there is a path $s, n_1, \ldots, d$
  - where each n_{i+1} is selected because
  - $\delta(n_{i+1}, d) = \min(\delta(n,d) : n \in N_{n_i})$

Non-greedy routable in $\mathbb{R}^2$:

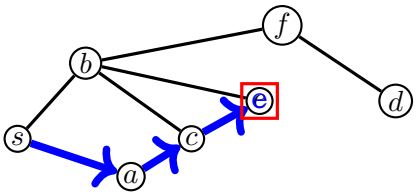


Every graph that contains $K_{1,6}$, i. e., a node with at least six neighbours that are not directly connected with each other, is not greedy-routable at this node.

The reason, why $K_{1,6}$ prevents a greedy embedding is, that you cannot distribute the six nodes in a way that the middle node is closer than the destination node, if you try to send a packet between adjacent neighbours. A solution to this problem is to prune the graph, but this risks separating the graph itself.

What happens is that the greedy routing algorithm runs into dead ends. And, there are other situations where greedy routing may fail.

Greedy Dead Ends



- Greedy route from source s to dest. d
- Nearest neighbour to d leads to
  - $a \to c \to e$
- Node e has no neighbour nearer to d
- Packet is stuck at e

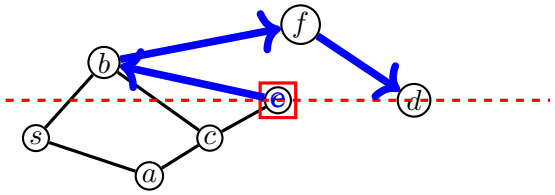
- Possible solution: switch to Face Routing

Similar for $compass$-Routing

One solution to handle dead ends is face routing. This means that a packet is routed around the dead end, at the surface (face) of the obstacle that makes up the dead end.

Face Routing

- Identify direct line to d
- Choose node closest to that line,
- Without crossing that line.
- Avoid loops (requires path history)



GOAFR+

This is a more efficient algorithm for face routing that will be explained in a next version of this lecture. See [**Kuhn03geometric**] for reference.

Finding a greedy embedding

- Not all graphs are greedy embeddable in all spaces
- Finding a greedy embedding is non-trivial
- Approach Wireless-Network
  - Positions are given by physical location
  - Often: too many neighbours
  - Local graph-pruning required
  - Mobility complicates matters
- Approach Self-organization
  - Given: communication graph
  - Node adopts local best position

- Popular: Rubberband-Heuristics, PSO
- See Kabourov 2012

5.3 Spring Embedder

Graph Embedding

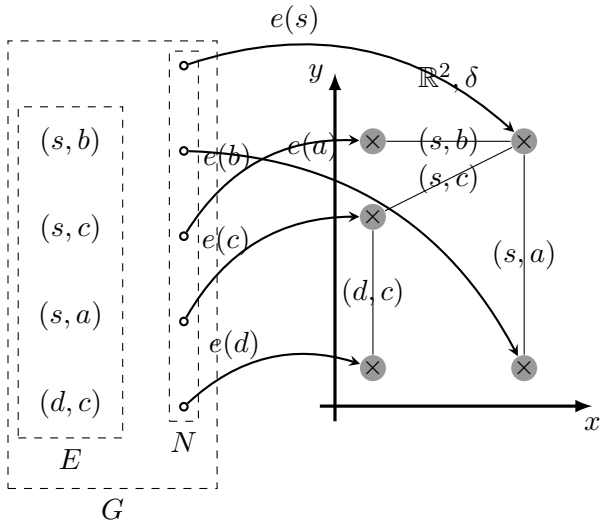


Figure 5.10: Embedding of a graph onto a metric space $\mathbb{R}^2, \delta$ by mapping each Node $n \in N$ onto a point. Edges then are added for clarity as lines between points.

- Graph: $G = (N, E), E \subseteq N^2$
- Embedding:
  - $e : N \to \mathbb{R}^n$
  - (more general: S with $\delta : S \times S \to \mathbb{R}$)

Ziele [Fruchtermann/Reingold 1991]:

1. Distribute the vertices evenly in the frame.
2. Minimize edge crossings.
3. Make edge lengths uniform.
4. Reflect inherent symmetry.
5. Conform to the frame.

Spring Embedder and Force Feed Algorithms

- Global graph Embedding using Fruchtermann/Rheingold:
- Attractive force: $f_a(d) = \frac{d^2}{k}$
- Repulsive force: $f_r(d) = \frac{-k^2}{d}$
- $k = C\sqrt{\frac{area}{|N|}}$ "space-per-node"
- Problem:
  - $|N|$ is a global value, only estimated
- Solution: Estimate!

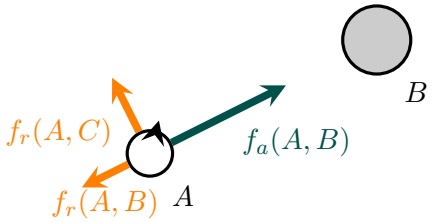


Figure 5.11: Force Feed of node A being attracted and repulsed by node B.

Embedding Algorithm

- Sum of repulsion to all nodes
- Sum of attraction to connected nodes

Difficult in distributed scenario!

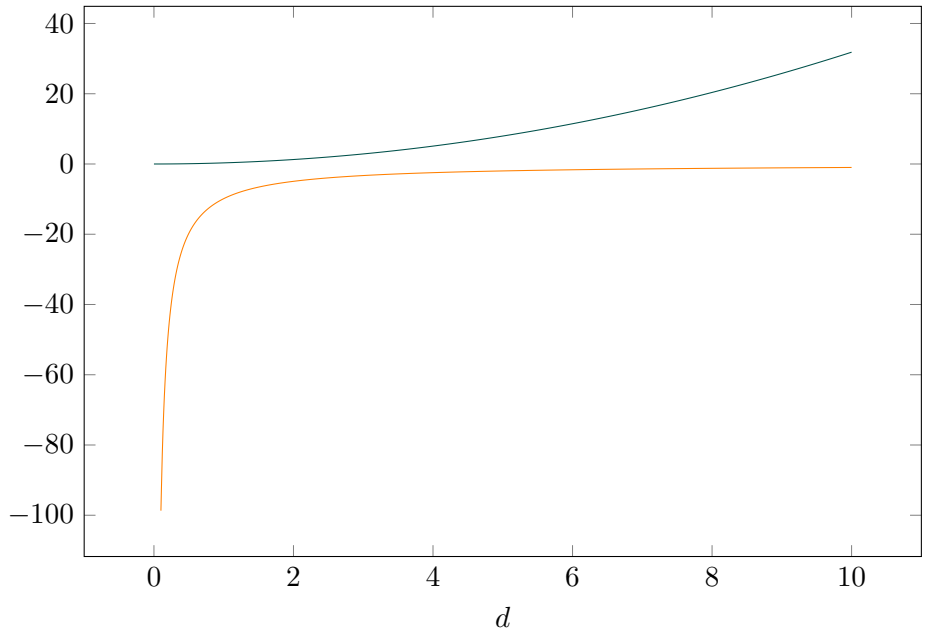


Figure 5.12: Function graph for example attractive and repulsive force ($k = \pi$)

6

Agent Communication

Dieser Abschnitt ist in wesentlichen Teilen dem Abschnitt Agentenkommunikation aus der Vorlesung von Prof. Dr. Astrid Niesse, Universität Oldenburg, Digitalisierte Energiesysteme, WiSe 22/23 entnommen.

Ziele und Hintergrund der FIPA-ACL [**Posland2007SpecifyingPF**]

Lernziele

- Verständnis der Herleitung von Performativen aus “speech acts”
- Kenntnis der FIPA und der Motivation hinter der FIPA
- Fähigkeit das Contract Net Protocol als Beispiel für ein komplexeres Interaktionsprotokoll darzustellen.

6.1 Speech Act Theory

Agentenkommunikation

- Sprachen wesentlich beeinflusst von speech act theory
- Basierend auf John Austin: How to do things with words
- Fragestellung:
  - Wie ereichen Menschen mit Sprache im Alltag ihre Ziele und setzen Intentionen um?
  - Welche Unterschiede gibt es?
  - Gibt es Charakteristiken in der sprachlichen Umsetzung?

- Sprache wird als Aktivität gesehen.

Die speech act theory beschreibt Sprachakte die nicht nur Sachverhalte oder Behauptungen darstellen, sonders selbst als Handlungen zu sehen sind. Damit werden Ziele und Intentionen eben nicht nur umschrieben, sondern der Sprachakt ist wesentlicher Teil der Umsetzung. Konkrete Beispiele sind zum Beispiel Äusserungen wie Eheschwüre. Aber auch das Stellen einer Frage, oder eine Antwort auf dieselbe stellen Handlungen in diesem Sinne dar. (Siehe auch die Wikipedia-Seite zu John Austin, Abschnitt How To Do Things With Words, zuletzt gesehen 2023-05-23)

> "It is, of course, not really correct that a sentence ever is a statement: rather, it is used in making a statement, and the statement itself is a 'logical construction' out of the makings of statements." –John Austin, How To Do Things with words, Lecture 1, Page 1

Der Begriff Illocutionary Acts wird von Austin als die Folge einer locution, also das was durch den Speech Act ausgelöst wird. Eine solche Aussage beschreibt, nach Austin, eben nicht nur eine Tatsache oder Behauptung, sondern ist performative in dem Sinne, als dass die Handlung des Sprechens den Zustand der Welt beeinflusst, i. e., verändert.

Illocutionary Acts (Searle, 1977)

Expressives Ausdrücke der Ansichten oder Emotionen der Sprechenden

- "Vielen Dank!"

Commissives Verpflichtungen auf zukünftiges Handeln des Sprechenden

- "Morgen werde ich das erledigen."

Directives Auslösung von Aktionen bei den Empfängern (Hörenden)

- "Kannst du mir die Butter geben?"

Representatives/Assertives Festlegung des Sprechenden auf die Wahrheit der ausgedrückten Tatsache

- "Es regnet."

Declarations Sprachhandlungen welche die Realität im Zusammenhang der Aussage ändern, e. g., Taufen, Kriegserklärungen, Eheformeln

- "Damit erkläre ich sie beide zu Mann und Frau."
- "Hiermit erklärt Land X Land Y den Krieg."

6.2 Agent Communication

6.2.1 Knowledge Query and Manipulation Language (KQML)

Agent Communication Languages

- KQML
- Rahmen für Agentenkommunikation
- Performatives: zur Umsetzung von Speech Acts
- `ask-if`: Nachfragen
- `perform`: bitte tue...
- `tell`: es ist...
- `reply`: Antwort auf Frage

```
(ask-one
   :content (PRICE A PRICE?)
   :receiver B
   :language C
   :ontology D
)
```

KQML

KQML

ein Auszug

Parameter	Bedeutung
`:content`	Inhalt d. Nachricht
`:language`	Formale Sprache d. Inhalt
`:ontology`	Definierte Semantik
`:reply-with`	Wird Antwort erwartet?
`:in-reply-to`	Worauf wird geantwortet?
`:sender`	Absender
`:receiver`	Empfänger

- KQML war unzureichend
- Weiterentwicklung zu Agent Communication Language (ACL) durch Foundation for Intelligent Physical Agents (FIPA)

6.2.2 Ontolgie

Der Begriff Ontologie stammt aus der Philosophie und beschreibt die Lehre von der Beschreibung der Welt. Allgemein wird der Begriff im Umfeld

der Informationswissenschaften inflationär für "Weltbeschreibungssprache" eingesetzt. Häufig sind damit Formalisierungen für die Zusammenhänge von semantischen Entitäten gemeint deren Ziel es ist alles darstellbar zu machen, was über die Welt gewusst werden kann.

6.2.3 Agent Communication Language (ACL)

FIPA

Foundation for Intelligent Physical Agents (FIPA)

- Standards für Agentenkommunikation
- Ziel: Interoperativität
- Teil der IEEE (seit 2005)
- Nicht mehr aktiv (Stand 2023)
- 28 Standards

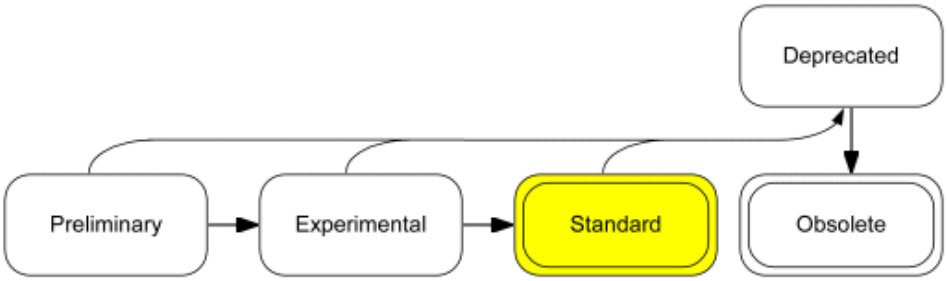


Abbildung 6.1: FIPA Standards Lifecycle, (URL), last 2023-05-23

Message Transport Reference Model

FIPA Agent Message Transport Reference Model

Message Transport Protocol (MTP) physischer Transfer zwischen Agent Communication Channels (ACCs)

Message Transport Service (MTS) Transportdienst des Agent Place (AP) mit dem ein Agent verbunden ist

ACL "Nutzlast" (payload) von MTP und MTS

6.2.4 Interaktionsprotokolle

FIPA-ACL

Identifier	Title
SC00001	FIPA Abstract Architecture Specification
SC00008	FIPA SL Content Language Specification
SI00014	FIPA Nomadic Application Support Specification
SC00023	FIPA Agent Management Specification
SC00026	FIPA Request Interaction Protocol Specification
SC00027	FIPA Query Interaction Protocol Specification
SC00028	FIPA Request When Interaction Protocol Specification
SC00029	FIPA Contract Net Interaction Protocol Specification
SC00030	FIPA Iterated Contract Net Interaction Protocol Specification
SC00033	FIPA Brokering Interaction Protocol Specification
SC00034	FIPA Recruiting Interaction Protocol Specification
SC00035	FIPA Subscribe Interaction Protocol Specification
SC00036	FIPA Propose Interaction Protocol Specification
SC00037	FIPA Communicative Act Library Specification
SC00061	FIPA ACL Message Structure Specification
SC00067	FIPA Agent Message Transport Service Specification
SC00069	FIPA ACL Message Representation in Bit-Efficient Specification
SC00070	FIPA ACL Message Representation in String Specification
SC00071	FIPA ACL Message Representation in XML Specification
SC00075	FIPA Agent Message Transport Protocol for IIOP Specification
SC00084	FIPA Agent Message Transport Protocol for HTTP Specification
SC00085	FIPA Agent Message Transport Envelope Representation in XML Specification
SC00088	FIPA Agent Message Transport Envelope Representation in Bit Efficient Specification
SI00091	FIPA Device Ontology Specification
SC00094	FIPA Quality of Service Specification
SC00097	FIPA Design Process Documentation Template

Abbildung 6.2: FIPA List of Standard Specifications, URL, last 2023-05-23

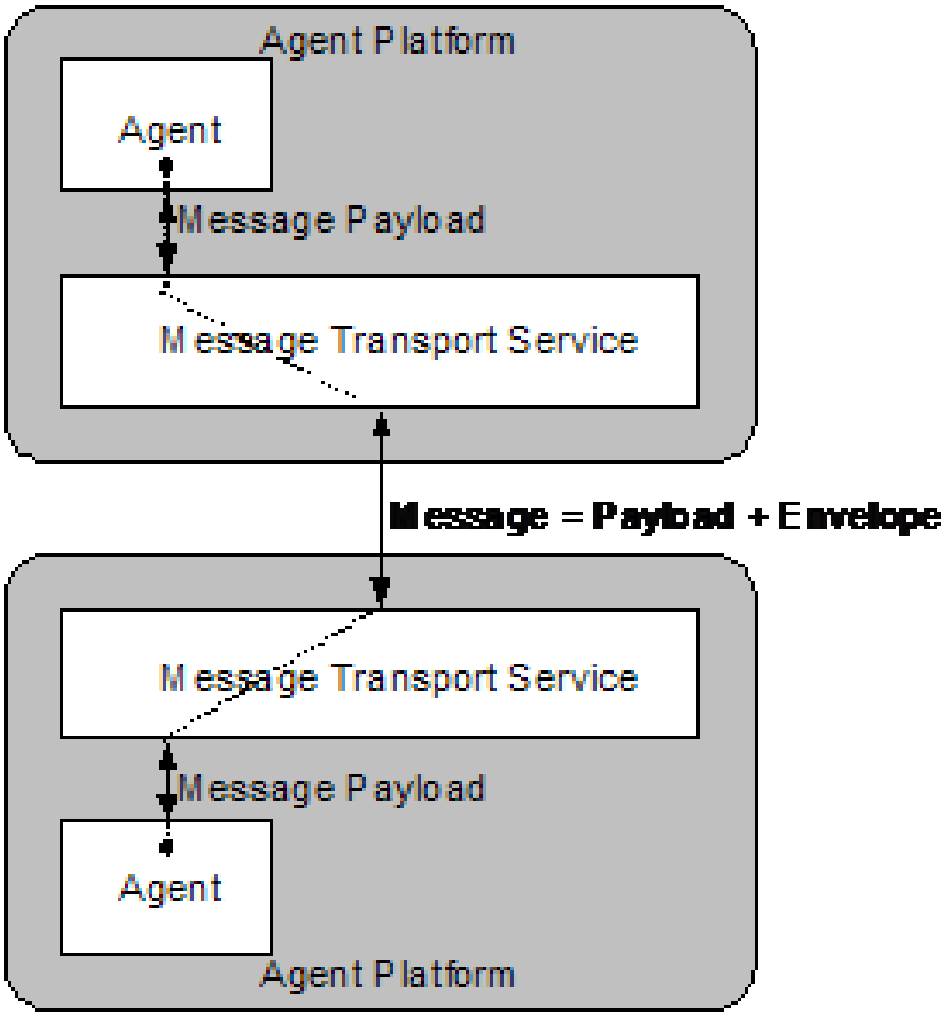


Abbildung 6.3: Message Transport Reference Model

- angelehnt an KQML
  - Request information
  - Passing information
  - Negotiation
  - Performing actions

Wichtig! Interaktionsprotokolle

- FIPARequest
- FIPAQuery
- FIPAContractNet
- FIPAAuctionDutch
- FIPAAuctionEnglish
- ...

FIPA-ACL Performatives

FIPA Performative Classification

FIPA Communicative Act Library Specification

FIPA Request Interaction

Die einfachste Interaktion ist eine directive.

FIPA Request Interaction

- Request to perform an action
- Participant decides
  - Accept, or
  - Refuse.
- Action result:

  failure indicate failure of action

  inform-done success indicator

  inform-result success indicator + results

performative	passing info	requesting info	negotiation	performing actions	error handling
accept-proposal			x		
agree				x	
cancel		x		x	
cfp			x		
confirm	x				
disconfirm	x				
failure					x
inform	x				
inform-if	x				
inform-ref	x				
not-understood					x
propose			x		
query-if		x			
query-ref		x			
refuse				x	
reject-proposal			x		
request				x	
request-when				x	
request-whenever				x	
subscribe		x			

Abbildung 6.4: FIPA Performatives Classification, http://www.fipa.org, last 2023-05-23

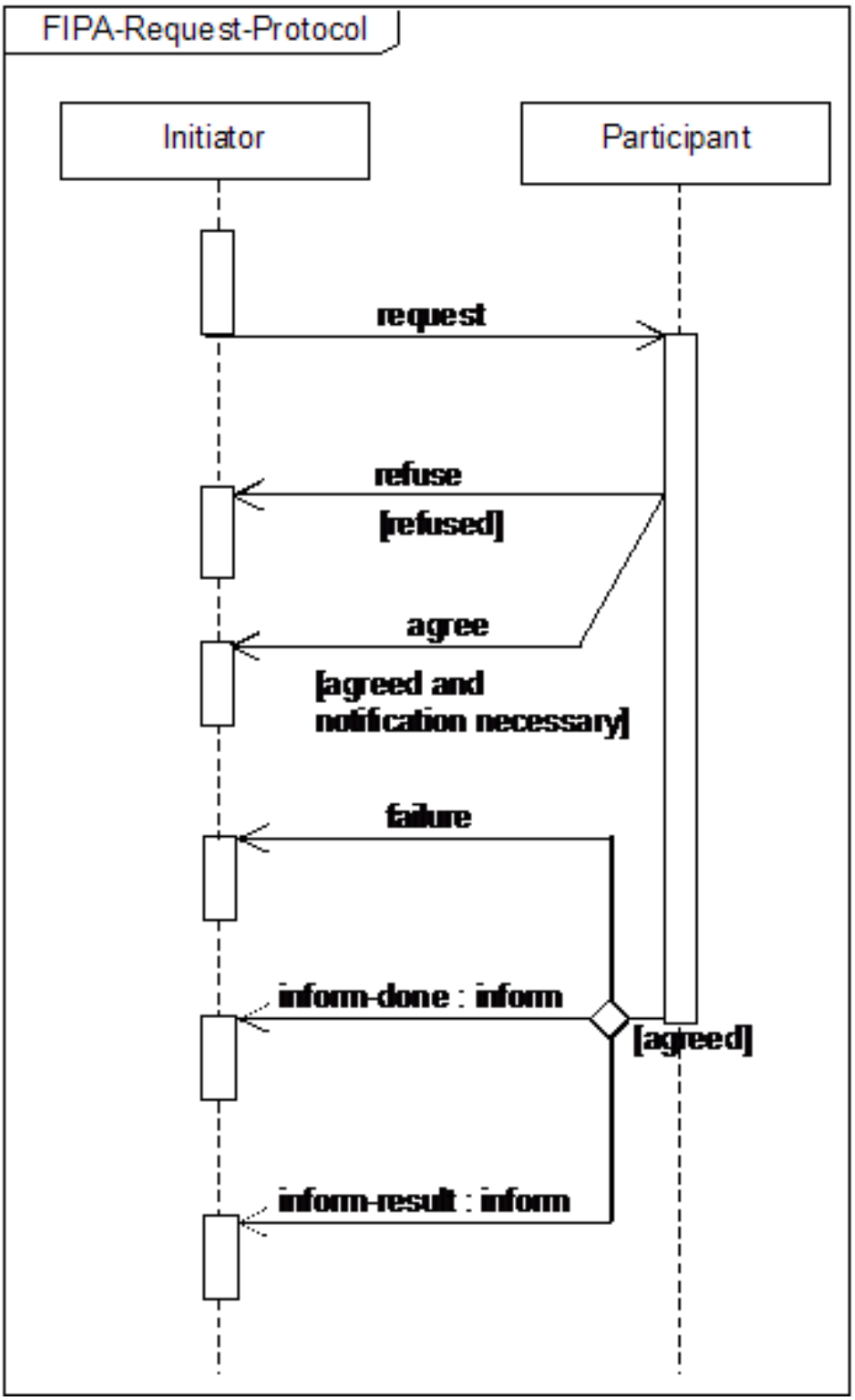


Abbildung 6.5: FIPA Request Interaction (Source, last 2023-05-23)

```
(request
  :sender (agent-identifier :name i)
  :receiver (set (agent-identifier :name j))
  :content
    "open \"db.txt\" for input"
  :language vb)
```

FIPA Contract Net Interaction

FIPA Contract Net Interaction

- Request with added proposal phase
- Sending m `cfp` to call in proposals
- Receiving n replies, of which
  - $i \leqslant n$ are `refuse`, and
  - $j = n - i$ are `propose`
- Initiator deciding and sending out
  - $k \leqslant n$ `reject-proposal`, and
  - $l = n - k$ `accept-proposal`.
- Receiving l Replies like for FIPARequest

FIPA Contract Net Example

```
(cfp
  :sender (agent-identifier :name j)
  :receiver (set (agent-identifier :name i))
  :content
    "((action (agent-identifier :name i)
      (sell plum 50))
     (any ?x (and (= (price plum) ?x) (< ?x 10))))"
  :ontology fruit-market
  :language fipa-sl)
```

6.2.5 FIPA Content Language

FIPA Content Language Specification

fipa00008

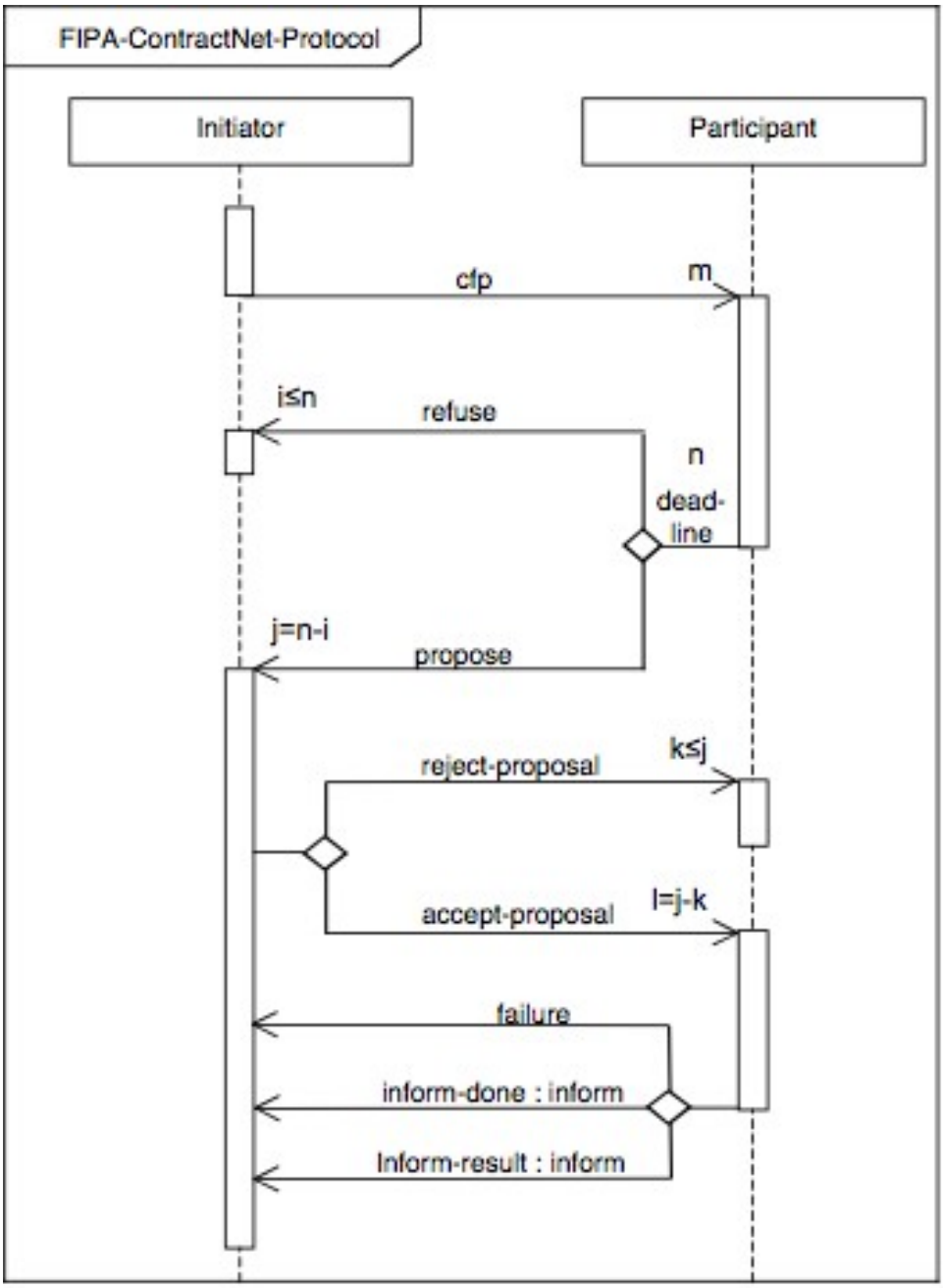


Abbildung 6.6: FIPA Contract Net Interaction, (see fipa00029, last 2023-05-23)

6.3 Mango Agent Communication

ACL in mango-agents

```
async def send_acl_message(self, content,
                           receiver_addr: Union[str, Tuple[str, int]], *,
                           receiver_id: Optional[str] = None,
                           acl_metadata: Optional[Dict[str, Any]] = None,
                           **kwargs) -> bool:
```

```
from mango.messages.message import Performatives
example_acl_metadata = {
    'performative': Performatives.inform,
    'sender_id': 'agent0',
    'sender_addr': ('localhost', 5555),
    'conversation_id': 'conversation01'
```

(see Mango Tutorial, Sending Messages, last 2023-05-24)

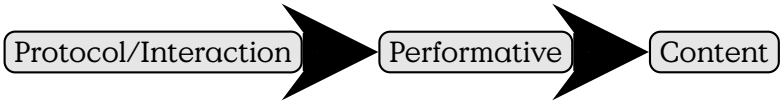
7

Semantic Language for Content Communication

Lernziele

- Motivation für standardisierte Content Languages
- Zusammenhang Content Language in FIPA Content Act Library
- Struktur der FIPA Semantic Language (SL)
- Formulierung von Well-formed-formulars und einfachen Queries

Recap



Performative with Content

```
(inform
   :sender (agent-identifier :name i)
   :receiver (set (agent-identifier :name j))
   :content
       "weather (today, raining)"
   :language Prolog)
```

[[**fipa00037**]]

FIPA Semantic Language

FIPA Semantic Language

content is

- Identifier of content (i. e., reference)
- Action expression
- Proposition

[[**fipa00008**]]

Well-formed formulas

Well-formed formulas

How to write statements on things?

We already know a few ways to formulate statements on things. For example using logic notation and predicates. This exactly is what we can use here. Additionally there are expressions on beliefs and uncertainty. Furthermore, as we are dealing with agents potentially capable of actions in the physical (or digital) world, we require statements on these actions.

unary `not`

binary `and`, `or`, `implies`, `equiv`

quantors `forall`, `exists` ∃

beliefs Belief/Uncertainty/Intention/Persistent Goal

actions `feasible`, `done`

Referential Operators

Referential Operators

iota exactly one match or failure

any any of fitting matches

all all (set) fitting matches

8

Game Theory

Lernziele

- Womit beschäftigt sich die Spieltheorie?
- Verbindung von Spieltheorie und Agentensystemen
- Kompetetive Agenten
- Adversarial Learning

Basierend auf der Vorlesung A.Niesse: Agentenbasierte Verfahren in Energiesystemen, Vorlesung 08, WiSe 22/23

8.1 Einstieg Spieltheorie

Die Idee der Spieltheorie, strategische Interaktionen zwischen rational handelnden Agenten, geht zurück auf John von Neumann.

Spieltheorie

- Fokus der Spieltheorie: Beschreibung und Analyse von Spielen
  - Spiel (allgemein): Wettbewerbssituation zwischen Beteiligten mit unterschiedlichen Handlungsoptionen
  - Wahl des einzelnen Spielers aus den Handlungsoptionen beeinflusst den Ausgang der Konfliktsituation
  - Damit immer: Entscheidungsfindung relevant für den jeweiligen Konflikt
- Spieltheorie gehört damit in den Bereich der Entscheidungstheorie

- Wesentliche Aspekte:
  - Rationalität der Entscheidungsfindung
  - Rückwirkung auf Wohlergehen
  - Abhängigkeiten zwischen Akteuren

8.1.1 Spiele

Züge und Spielpläne

Zug Einzelentscheidung eines Spielers in einem Zeitschritt eines Spiels

Spielplan beschreibt die Auswahl von einem oder mehreren Zügen abhängig vom Zustand des Spiels

- Wenn [Zustand des Spiels über vorangegangene Züge] ... dann Zug Y

Strategie Vollständig vorliegender Spielplan vor Beginn des Spiels

vollständig für jeden möglichen Zustand des Spiels gibt es eine Angabe zum auszuwählenden Zug.

Utility und Spieltheory

Utility Stellt eine Ordnung über den Handlungsoptionen der Spieler her.

Utility function Bildet realweltliche Zustände (outcomes), die sich aus der Auswahl einer Handlungsoption ergeben, auf einen Wert ab.

Unsicherheit wird über eine erwartete Utility abgebildet (expected utility)

Spiele im Sinne der Spieltheorie sind Interaktionen zwischen mehreren utility- bezogenen Spielern

Mit dieser Sichtweise lassen sich sehr viele realweltliche Probleme spieltheoretisch modellieren – Überführung von realweltlichen Zuständen in die mathematische Betrachtung

8.2 Beispiel Abendgestaltung

Stellen wir uns vor Alice (A) muss sich entscheiden ob wie sie den Abend gestalten will. Grundsätzlich kann Sie sich zwischen den folgenden Optionen entscheiden.

Beispiel: Abendgestaltung

Optionen:

- Club (c)
- Movie (m)
- Home (h)

Zuerst einmal darf Alice sich überlegen, auf welche der Aktivitäten sie selbst Lust hat. Diese Single-view-utility stellt die Situation ohne Berücksichtigung weiterer Faktoren dar. Dies wird dadurch dargestellt, dass jeder Auswahl ein Utility-Wert zugeordnet wird. Ein höherer Wert stellt dabei eine höhere Utility dar, ist also besser bewertet.

Utility Alice:

- $c \mapsto 100, m \mapsto 50, h \mapsto 50$
- (Single-view utility)

Natürlich ist das Leben nicht ganz so einfach, denn Alice ist sich bewußt, dass auch andere Personen ihren Abend planen. Und aus persönlichen Gründen möchte Alice ungerne auf Bob treffen. Ihr wäre es schon unangenehm mit ihm im Kino zusammenzutreffen, aber einen Clubbesuch würde ihr Bob schon sehr massiv verhageln. Sie bewertet dies als Malus auf die Single-view utility.

Alice bezüglich Bob

- Bob im Kino treffen: -40
- Bob im Club treffen: -90

Über Carol würde sich Alice freuen. Egal wo, wenn Carol dabei wäre, hätte Alice nochmal halbsoviel Spaß wie ohne Carol.

Alice bezüglich Carol:

- Utility $\cdot 1,5$

Wesentlich ist, bei der Auswertung, die Reichenfolge der Operationen. In diesem Fall berechnen wir zuerst den Malus durch die Anwesenheit von Bob und anschließend gegebenenfalls den Boost durch die Anwesenheit von Carol.

Alice Baseline Utility:

- Staying Home: 50

Utility Funktion bewerten: siehe Abb **??** und Abb. **??**.

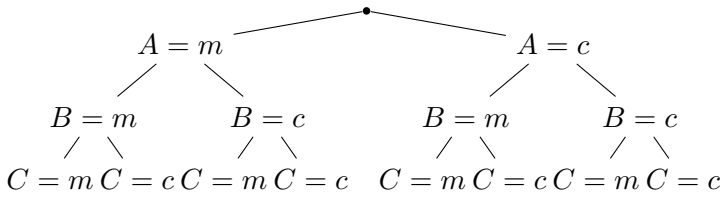
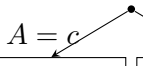


Abbildung 8.1: Ereignisbaum für das Beispiel Abendgestaltung



	$B = c$	$B = m$
$C = c$	15	150
$C = m$	10	100

	$B = c$	$B = m$
$C = c$	50	10
$C = m$	75	15

Abbildung 8.2: Basic Utility Values für verschiedene Ausgänge des Experiments

Die Utility-Funktion liefert uns für jedes mögliche Ergebnis eine Bewertung. Damit lässt sich feststellen, dass der beste Ausgang für Alice wäre, wenn sie und Carol im Club tanzen gehen und Bob sich für das Kino entscheidet. Dieses Ergebnis kann Alice benutzen um das gewünschte Ergebnis, e. g., durch Beeinflussung des Verhaltens von Bob und Carol herbeizuführen.

In der Spieltheorie — und in der Realität — kann Alice aber nur Entscheidungen für sich selbst treffen. Das bedeutet, Alice kann sich nur zwischen ihren drei Optionen entscheiden. Entscheidet sie sich zuhause zu bleiben, dann kann sie sicher sein wie der Abend ausgeht. Entscheidet sie sich allerdings anders riskiert sie entweder einen katastrophalen Abend, kann aber auch das oben gefundene Optimum erreichen.

Für solche Situationen kennt die Stochastik den Erwartungswert. Abhängig von der Wahrscheinlichkeit einzelner Ereignisse lässt sich berechnen welche "Belohnung" sie für welche Entscheidung erhoffen kann. Damit lässt sich genauer bewerten wie oft eine Entscheidung besser oder schlechter als eine andere ist.

Erwartungswert

- Expected utility: gewichtetes Mittel Eintrittswahrscheinlichkeiten
  - $P(B = c) = 0,6$; $P(B = m) = 0,4$
  - $P(C = c) = 0,25$; $P(C = m) = 0,75$

- Option Club $E_c = 51,75$

$$0,25 \cdot (0,6 \cdot 15 + 0,4 \cdot 150) + 0,75 \cdot (0,6 \cdot 10 + 0,4 \cdot 100)$$

- Option Movies $E_m = 46,75$

$$0,25 \cdot (0,6 \cdot 50 + 0,4 \cdot 10) + 0,75 \cdot (0,6 \cdot 75 + 0,4 \cdot 15)$$

- Option Home $E_h = 50$

Das Ergebnis dieses Beispiels ist denkbar knapp. Verschiebungen in den Wahrscheinlichkeiten oder den Bewertungen können dazu führen, dass das Ergebnis sich umdreht.

Dieses Beispiel ist ein Spiel mit nur einer Runde, das bedeutet die Spieler haben wenig Gelegenheit zu interagieren. Anders sieht die Situation aus, wenn ein solches Spiel über mehr als eine Runde läuft. Wie sieht das Ergebnis auf längere Sicht aus, wenn alle Akteure ihr Verhalten anpassen können. Bob könnte zum Beispiel versuchen Alice häufiger zu treffen und sich bewußt die Orte aussuchen die Alice in der Vergangenheit häufiger aufgesucht hat. Carol könnte es nach Abwechslung verlangen, was die Wahrscheinlichkeiten ihres Verhaltens von Runde zu Runde umdrehen könnte.

Insbesondere, wenn die Spieler entgegengesetzte Interessen verfolgen sind Spiele über mehrere Runden interessant und manchmal überraschend.

8.2.1 Gefangenendilemma

Gefangenendilemma

Regeln:

1. Beide kooperieren, 3 Punkte
2. A kooperiert, B schweigt: A 0, B 4
3. A schweigt, B kooperiert: A 4, B 0
4. Beide schweigen: beide 2

8.3 Reinforcement Learning

This is a very brief and cursory introduction. The objective is only to provide you with sufficient information that you can understand the basic idea. For the start remember the idea of a learning agent from Section **??**.

One fundamental idea is to “learn from your mistakes and successes” and to motivate repeating successful solutions and discourage failed ways of solving a task.

OpenAI Hide and Seek

Re-inforcement learning

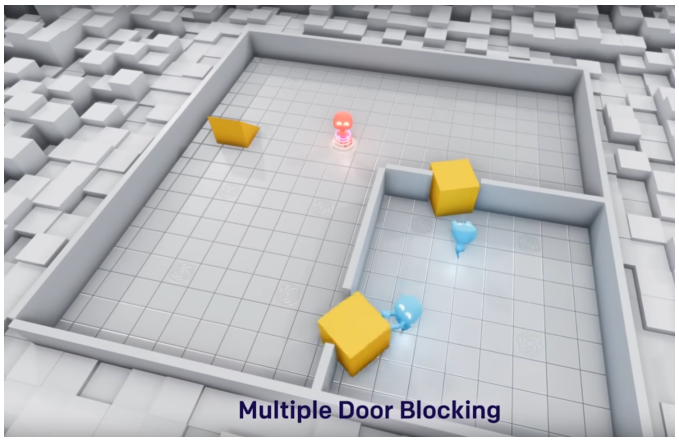


Figure 8.3: OpenAI Hide and Seek Agents, 2019

A paper-release video can be found here OpenAI Hide and Seek Agents, 2019.

9

Mobile Agents

Lernziele

- Elemente eines mobilen Agenten
- Welche Schritte umfasst ein Migrationsprozess
- Spezifische Probleme von mobilen Agenten

Based on [**BraunRossack05**]

Terminology

agency mobile agent server

agent place subdivision of agency

mobile agent system network of agencies

agent toolkit specific agent system

Mobile Agent

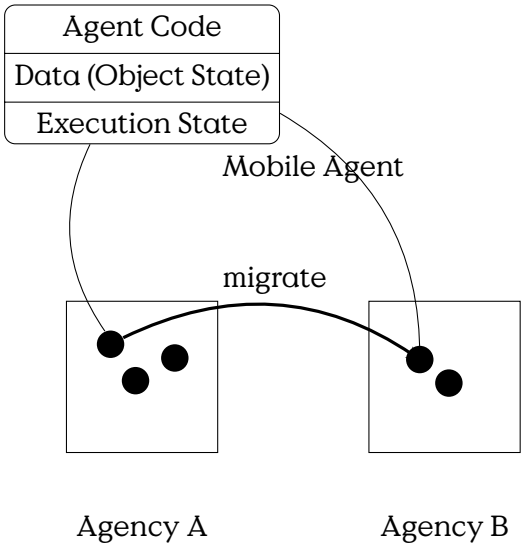
Code Agent Programm

- distinguished from Agency code/libraries
- specifies behaviour

Data (Object State)

- Instance variables
- “Payload”

Execution State
- State not handled by agent code
- e. g., Process Counter



Mobile Agent Platforms

- SeMoA (Secure Mobile Agents)
  - unmaintained since 2007
  - Research Project
  - Fraunhofer IGD (Darmstadt)
- Java Agent DEvelopment Framework (JADE:
  - discontinued 2011
- AgentLink III (last 2005)
- Mobile-C
  - latest news 2019 (That's almost alive)
  - C/C++ Agents
  - FIPA compliant
  - Requires Embedded Ch Toolkit (proprietary)

Agent Migration Process

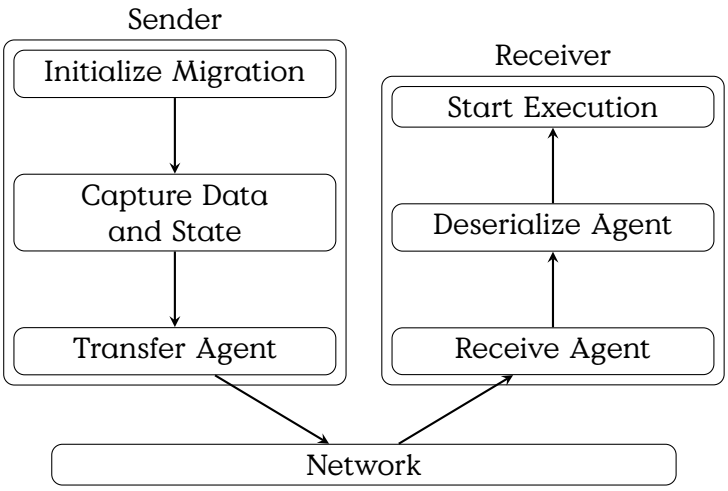


Figure 9.1: Schematic Agent Migration Process [**BraunRossack05**]

Specific Problems for Migrating Agents

- Local Agent Identifier (mango: `aid`)
- Trustworthyness of results
- Serialization of Agent State
  - Requires agency support
  - Stopping/Restarting vs. Execution State
- Programming/Interpreter Language

Mango Migration for ACO

- Create specific serialisation method for state
- Code encoding?
- Build migration agency Agent (as TCP endpoint)
  - Now!

10

Learning Agents

Lernziele

In diesem Abschnitt möchte ich einen kleinen Einstieg in lernende Systeme geben damit wir verstehen was “lernende Agenten” sind.

Lernziele

- Einfluss der externen Performance verstehen
- Grundverständnis der Funktionsweise “lernender” Maschinen
- Fähigkeit eine physische lernende Maschine bauen zu können
- Fundamentale Grundlagen neuronale Netze (Perceptron)
  - Inputs und Outputs
  - Wissensrepräsentation im Perceptron
  - Entscheidungsfunktion

10.1 Machine Educable Noughts And Crosses Engine (MENACE)

MENACE

Machine Educable Noughts And Crosses Engine

- Donald Michie: Trial and Error [**michie1961trial**]
- Matchbox learning explained [**gardner1969matchbox**]

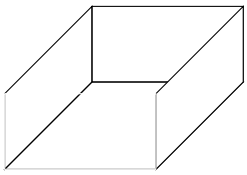
MENACE

- Coloured beads in boxes
  - Colour represents next move
  - Box represents specific state
- Next Move
  - Shake box
  - Randomly draw bead
- Learning: each box modify chosen color,

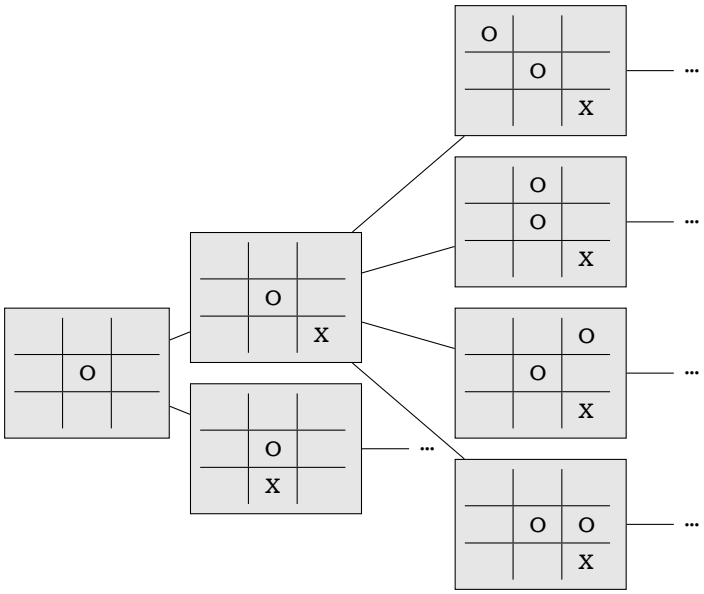
  WIN add 3 beads

  DRAW add 1 bead

  LOSE remove bead



Noughts and Crosses



To build MENACE you require one box per potential state of the game. You can optimize by removing rotations and mirrored states as those are

equal for the purpose of the game[1]. With optimizations you can build a working machine with 304 Boxes[2].

MENACE simulator (online)

10.2 Artificial Neuronal Networks

10.2.1 Perceptron

10.3 Learning

10.3.1 Gradient Decent

10.4 Reinforcement Learning

One fundamental idea is to "learn from your mistakes and successes" and to motivate repeating successful solutions and discourage failed ways of solving a task.

[1] although keeping these states might lead to longer training and different reactions of the machine depending on the particular branch

[2] The calculation of minimum box number is almost more complicated than running MENACE.

11

Mobile Agent Security

12

Metaheuristics