# Situativ-episodisches Schliessen bei Wolf und Schafe

Hansruedi Kaiser Juni 2003

## 1 Ganzheitliches Erkennen

Grundsätzlich geht das ILM (integrierendes Lernmodell) davon aus, dass der Prozess des Erinnerns auf situativ-episodische Ebene irgendwie „ganzheitlich“ erfolgt, d.h. das die aktuelle Situation nicht zuerst irgendwie in Merkmale zerlegt und dann anhand dieser Beschreibung mit Inhalten im Gedächtnis verglichen wird. Durch die Beschreibung anhand von Merkmalen würde bereits eine Verschiebung auf die deklarative Ebene geschehen.

Bei DEPP war es möglich, dies in der Simulation zu realisieren, indem DEPP wahrnehmungsmässig vollständig identische Situationen als identisch erkannte und jede solche Situation als ein Objekt ohne Eigenschaften gespeichert wurde. DEPP war deshalb aber auch nicht in der Lage, Ähnlichkeiten zwischen wahrnehmungsmässig nicht identischen Situationen zu erkennen.

Prinzipiell wäre es möglich, auch bei den Schafen so vorzugehen. Beschränkt man ihr Geschichtsfeld auf die nähere Umgebung, so ist die Anzahl wahrnehmungsmässig unterscheidbarer Situationen etwa in derselben Grössenordnung wie beim DEPP und damit in einem Simulationsprogramm durchaus handhabbar. Verschiedene Gründe sprechen aber gegen ein solches Vorgehen:

- Das ILM geht nicht davon aus, dass die Einheiten des situativ-episodischen Wissens die wahrnehmungsmässig unterscheidbaren Situationen sind. Dies macht schon deshalb kein Sinn, da sich für Lernende in einer etwas komplexeren Umgebung dieselbe wahrnehmungsmässig identische Situation kaum zwei mal ergibt.
- Für DEPP ergab sich aus dieser aus dieser Anordnung das Problem, dass verschiedene Umweltkonstellationen als dieselbe Situation wahrgenommen wurden, obwohl sie ganz andere Eigenschaften hatten. Dies war bei DEPP in Ordnung, da wir unter anderem am Studium genau dieser Probleme interessiert waren. Bei den Schafen ist dies nicht oder nur eingeschränkt der Fall.
- Im Gegensatz zu DEPP stehen die Schafe mit anderen Schafen in Kontakt und tauschen Informationen aus. Diese Gespräche führen einerseits dazu, dass das Schaf die Wahrnehmung der aktuellen Situation über seine primäre Wahrnehmung hinaus ausdehnen kann, und andererseits sind sie selbst natürlich Teil der "Situation". Dadurch wird die Umwelt viel komplexer (und realistischer) und wahrnehmungsmässig völlig identische Situationen dürften sehr selten sein.
- Und nicht zuletzt sollen die Schafe explizit in die Lage versetzt werden, aus Erinnerungen auf nicht wahrnehmungsmässig identische Situationen Rückschlüsse zu ziehen, bzw. überhaupt Ähnlichkeiten zwischen der aktuellen Situation und beliebigen Erinnerungen erkennen zu können.

Damit dies im Rahmen einer Computersimulation möglich ist, müssen die Einheiten des situativ-episodischen Wissens der Schafe irgendwie eine interne Struktur haben. Im Folgenden geht es darum, eine geeignete solche Struktur zu finden.

Diese Struktur wird sich als Folge der Implementation als Computerprogramm notwendigerweise auch als deklaratives Wissen der Schafe interpretieren lassen. Mögliche

Schlüsse, die sich daraus ergeben, sind aber mit Vorsicht zu behandeln, da es sich auch um eine negative Analogie dies Modells handeln könnte.

# 2 Drei Modelle des Abrufs analoger Erinnerungen

## 2.1 Neuronale Netze

In neuronalen Netzen wird das Wissen in Form von mehr oder weniger starken "aktivierungsleitenden Verbindungen" gespeichert. Eine Möglichkeit, diese zu nutzen, besteht darin, dass die Wahrnehmung einer neuen Umweltkonstellation als eine Reihe mehr oder weniger aktiver "Knoten" dargestellt wird. Diese Knoten können für die Aufgabe Eigenschaften der Umweltkonstellation sein (z.B. "direkt vor mir steht der Wolf": +1 = ja, -1 = nein), sie können aber auch einfache Wahrnehmungsprimitiva darstellen. Die Aktivierung dieser Knoten breitet sich dann über Verbindungen auf andere Konten aus, bis schliesslich eine zweite Reihe von Knoten erreicht wird, an deren Aktivierungszustand das Ergebnis (z.B. die sinnvollste Aktion) abgelesen werden kann.

Ähnlichkeiten zwischen verschieden Umweltkonstellationen werden dadurch erkannt, dass bei einem richtig eingestellten neuronalen Netz ähnliche Inputaktivierungen zu ähnlichen (bzw. innerhalb bestimmter Schwankungsgrenzen gleichen) Outputaktivierungen führen. Was in diesem Sinn als "ähnlich" gelten soll, wird gelernt, indem die Verbindungen "trainiert" werden, bis sie bei den unterschiedlichen typischen Inputs den gewünschten Output zeigen. Das Wissen um die Ähnlichkeit wird so auf eine nicht offensichtliche Art in den mehr oder weniger starken Verbindungen zwischen einzelnen Knoten gespeichert.

## 2.2 Indizierung („klassisches CBR“)

Im Gegensatz dazu wird die Ähnlichkeit bei der Form des analogen Erinnerns, wie sie in der klassischen Künstlichen Intelligenz unter dem Begriff "case based reasoning" bekannt ist, wesentlich direkter realisiert. Sämtliche gespeicherten Fälle werden durch denselben Satz von Eigenschaften ("slots") beschreiben, die je eine vordefinierte Menge von Werten ("slot filler") annehmen können. Dies entspricht in etwa den Inputknoten eines neuronalen Netzes.

Zu jedem Fall wird aber auch direkt der entsprechende Output gespeichert, d.h. etwa festgehalten, welche Aktion in diesem Fall sinnvoll ist. Tritt nun eine neue Situation auf, wird in der Erinnerung nach ähnlichen Fällen gesucht, indem die Fälle herausgegriffen werden, die in möglichst vielen Eigenschaften mit der neuen Situation übereinstimmen.

Damit nicht nur identische gefüllte Slots für die Feststellung einer Ähnlichkeit herbeigezogen werden können, werden zudem meist noch Ähnlichkeitsmasse für die möglichen Füllungen definiert. Sind dies numerische Werte, können das Intervalle sein. Bei Begriffen werden oft Abstraktionshierarchien verwendet ("Wolf" und "Schaf" sind beides "Mitspieler"). Ähnlichkeit wird hier also "deklarativer" definiert als bei den neuronalen Netzen.

## 2.3 Strukturvergleiche („muliple constraint satisfaction“)

Noch weiter gehen Modelle wie SME ([Falkenhainer, 1989 #3354]) und ACME [Holyoak, 1996 #3328]. Hier wird der Fall "vollständig deklarativ" als komplexe Struktur von Prädikaten dargestellt. D.h. er wird nicht nur einfach Aneinanderreihung von Eigenschaften repräsentiert, sondern die Beziehungen der einzelnen Slots untereinander werden ebenfalls dargestellt (z.B. <Feld1=Wolf, Feld2=Schaf, blockiert(Feld1, Feld2)>. Wird ein zur aktuellen Situation analoger Fall gesucht, werden sowohl die Struktur wie die Füllungen der Slots verglichen. Dadurch ist es möglich, Fälle als ähnlich zu erkennen, die zwar in den Füllungen der Slots nicht direkt übereinstimmen, in denen aber diese Füllungen z.B. dieselbe Funktion übernehmen (wie etwa in <Feld1=Wolf, Feld2=Schaf, blockiert(Feld1, Feld2)> und <Feld1=Schaf, Feld2=Schaf, blockiert(Feld1, Feld2)>).

Dieser Vergleich kann mehr oder weniger rigide erfolgen. SME vergleicht nur die Struktur, berücksichtigt die Inhalte der Slots also nicht. Es stellt immer 1:1 Beziehungen zwischen identischen Beziehungen her. ACME ist flexibler. Hier können z.B. auch nicht identische Beziehungen aufgrund ähnlicher Slotinhalte zueinander in Beziehung gesetzt werden. (Interessant ist, dass ACME für den Prozess, die bestmögliche Beziehung zwischen zwei Fällen zu finden, auf Techniken der Neuronalen Netze zurückgreift!)

# 3 Wahl eines geeigneten Modells für die Schafe

Das ILM geht davon aus, dass die Elemente des situativ-episodischen Wissens bewusstseinsfähig sind, d.h. dass es möglich ist, einzelne, erinnerte Situationen in Form von Geschichten zu beschreiben. Dies dürfte bei einer Simulation in Form eines neuronalen Netzes schwierig zu erreichen sein. Nicht für die aktuelle, auslösende Situation, denn diese muss ja in Form von Aktivierungen der Knoten des Input-Layers des Netzwerkes kodiert werden. Und selbstverständlich ist es möglich, diese Knoten und ihre Aktivierungen so zu wählen, dass sich an ihnen sozusagen die „Geschichte“ ablesen lässt. Das ILM geht aber im weiteren davon aus, dass dieser „Input“ an meist mehrere ähnliche Situationen erinnert und dass diese Erinnerungen zusammen dann zum „Output“ führen. Zumindest einige dieser Erinnerungen werden bewusst und die entsprechenden Geschichten können erzählt werden. Bei einem neuronalen Netzwerk sind diese Zwischenschritte implizit im Netz durch die Gewichte der Verbindungen und Aktivierungen von „hidden layers“ (Mengen von Knoten, die zwischen dem Input und dem Output liegen) gegeben. Es dürfte schwierig sein, diese Zwischenschritte als Erinnerungen an konkrete Situationen aus dem Netz zu lesen (wenn dies auch eine interessante Aufgabe wäre). Aus diesem Grund werde ich – zumindest vorläufig – kein neuronales Netz einsetzen.

Die beiden anderen Varianten sind nicht grundsätzlich voneinander verschieden. Auch beim Strukturvergleich müssen zuerst aus allen vorhandenen Fällen einige Kandidaten heraus gefiltert werden, die dann weiter bearbeitet werden können. Und dieser Filtervorgang muss sich auf eine Art Index stützen, über den in der Fallbasis nach Fällen gesucht wird, die bezüglich gewisser Merkmale gleiche oder ähnliche Werte haben, wie der aktuelle Fall. Der wesentliche Unterschied liegt darin, dass beim klassisches CBR die Fälle alle eine identische Struktur haben. Es ist von vorneherein bekannt, welcher Slot des einen Falles welchem Slot des anderen Falles entspricht und Ähnlichkeit kann einfach dadurch bewertet werden, dass man die Ähnlichkeit der Inhalte der korrespondierenden Slots bewertet. Beim Strukturvergleich gibt es eine derartige Einschränkung nicht, d.h. die einzelnen Fälle können sehr unterschiedlich strukturiert sein und entsprechend muss in einer zweiten Phase zusätzlich beurteilt werden, ob die in der ersten Phase gefundenen ähnlichen Merkmale auch in einem ähnlichen strukturellen Bezug zueinander stehen.

Die klassische CBR Variante funktioniert gut, wenn die ganze Fallbasis im Hinblick auf einen bestimmten Verwendungszweck angelegt wird. Dann ist es möglich, die Struktur optimal auf diesen Zweck hin zuzuschneiden. Sie versagt aber, wenn die so abgelegten Fälle in einem ganz anderen Kontext gebraucht werden sollen. Dann ist ein flexibles Vorgehen entsprechend dem Strukturvergleich notwendig.

Im Rahmen von Wolf und Schafe dürften beide Varianten von Bedeutung sein. Neue Situationen, welche die Schafe während des Spielens in ihr Gedächtnis aufnehmen, werden von ihnen im Rahmen der Simulation auch nur während des Spiels weiter verwendet. Sie können entsprechend auf diese Verwendung hin optimiert abgespeichert und nach klassischem CBR verwendet werden. Neulinge hingegen, also Schafe, die noch nie oder nur wenig mitgespielt haben, müssen in der Lage sein, ihr Vorwissen, ihre Erinnerungen ans Situationen ausserhalb des Spiels zu mobilisieren. Sie müssen flexibler Analogien herstellen können entsprechend dem Strukturvergleich. In eine ähnlichen Lage dürften Schafe sein, die auf eine neue Position kommen, auf der sie bisher noch nicht gespielt haben.

Vom Ablauf her könnte man sich also grob vorstellen, dass jedes Schaf zuerst versucht mittels eines CBR Vorgehens direkt aus Erfahrungen mit Spielsituationen zu schöpfen.

Erweist sich dies als unergiebig, wird der Fokus ausgeweitet und es kommen Strukturvergleiche zum Zuge.

Interessant wird allerdings der Übergang vom ganz neuen Schaf zum etwas erfahrernen Schaf sein, da dieses zuerst das geeignete Format zum Abspeichern der neuen Situationen erlernen muss. Damit dies geschieht, muss es generell das Ziel haben, eine solches neues, optimiertes Format zu entwickeln. Bei der konkreten Ausgestaltung dürfte dann das Gespräch mit den anderen Schafen und die in deren "Fachsprache" verwendeten Begriffe weiterhelfen.

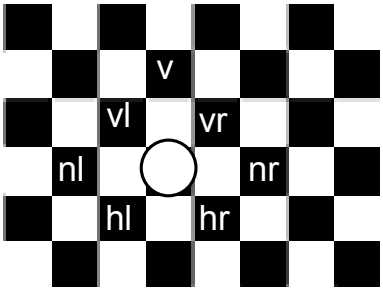
Zu erwarten ist zudem, dass Situationen, die während dieser Übergangsphase gespeichert werden, nachher im CBR Modus, d.h. im Spielalltag nicht mehr zu Verfügung stehen.

# 4 Ein erster Entwurf einer einfachen Sitautionsrepräsentation erfahrener Schafe

## 4.1 Eigenschaften der Situationen

- Rolle: In welcher Position habe ich das Spiel begonnen? (s1, s2, s3, s4 oder s5)

- Direkt vor
- Vorne links
- Vorne rechts
- Neben links
- Neben rechts
- Hinten links
- Hinten rechts




(mögliche Werte jeweils: schaf, wolf, wand oder leer)

- Distanz zum Wolf (direkt_vor (wenn vl oder vr), weiter_weg)
- Schritt (was tut das Schaf: links, rechts, stehenbleiben)

Dies genügt im wesentlichen, damit das Schaf sein eigenes Vorgehen als einfache Reaktion steuern kann. Ob es genügt, um einerseits mit den anderen Schafen zusammen zu entschieden, was wirklich getan werden soll, und um andererseits Neulinge anzuleiten, muss sich zeigen (vgl. .

## 4.2 Bewertung der Fälle

Jeder Fall illustriert ein mehr oder weniger geglücktes Vorgehen. Jeder Fall wird deshalb auf einer Skala von -1.0 (wenn immer möglich vermeiden) bis 1.0 (wenn immer möglich anwenden) bewertet.

## 4.3 Aufruf der Fälle

Gesucht wird in allen vorhandenen Fällen die vier (?) Fälle, bei dem die meisten Eigenschaften dieselben Werte haben wie die aktuelle Situation. Da die Fallbasis nicht sehr gross sein wird (maximal 81920 Fälle) ist das vielleicht sogar technisch ziemlich direkt machbar.

## 4.4 Verwendung der Fälle

Der Vorschlag für das weitere Vorgehen des Schafes ergibt sich direkt aus dem Wert der Eigenschaft "Schritt". Es sind keine Anpassungen an die konkrete Situation notwendig.

Einzige Ausnahme entsteht, wenn das Feld, auf welches das Schaf vorrücken soll, besetzt ist. Dies ist durchaus denkbar, wenn die aktuelle Situation einem gespeicherten Fall in allen Punkten sehr ähnlich ist, ausser dass das entsprechende Feld nicht leer sondern von einem anderen Schaf besetzt ist. Dann kann dieser Fall zum am besten passenden Fall werden, obwohl er nicht anwendbar ist. Tritt dies ein, muss eine geeignete (deklarative?) Stoppregel dafür sorgen, dass der Fall verworfen und der nächst bessere gewählt wird.

Aus den vier Fällen wird dann für jede der drei möglichen Aktionen ein Wert berechnet, der aus der Summe über alle vier Fälle der Produkte von aktion_durchgeführt(1 oder 0) * Wert des Falles * Ähnlichkeitswert besteht. Die Aktion, die den höchsten Wert erhält, wird vorgeschlagen.

## 4.5 Speicherung neuer Fälle

Grundsätzlich wird jede erlebte Situation als neuer Fall gespeichert. Zu lösen wäre die Frage, was geschieht, wenn identische Situationen auftreten.

## 4.6 Nachführung der Bewertung der Fälle

Für einige Fälle ist dies einfach:

- Klar positiv, wenn die Schafe durch die Aktion gewinnen
- Klar negativ, wenn die Schafe durch die Aktion verlieren
- Positiv bzw. negativ, wenn andere, erfahrene Schafe das sagen

In allen anderen Fällen muss wohl eine Rückwärtsverkettung der Bewertungen vorgenommen werden:

- Positiv, wenn die Aktion in eine Situation führt, die (in Kombination mit einer entsprechenden Anschlussaktion) bereits als positiv bewertet ist
- Negativ, wenn die Aktion in eine Situation führt, aus der nur negativ bewertete weiterführende Aktionen bekannt sind.

## 4.7 Ein Implementationsbeispiel

In CASL (www.aber.ac.uk/compsci/Research/mbsg/cbrprojects/CASL1pt3.pdf) sieht das dann etwa wie folgt aus.

**Eigenschaften, die man eingeben muss**

Die Gewichte legen das Gewicht fest, dass eine Übereinstimmung zweier Fälle bezüglich einer bestimmten Eigenschaft hat.

```
field sn type is (s1, s2, s3, s4, s5)
      prompt is ['welches Schaf:'];
field vl type is (schaf, wolf, wand, leer)
      weight is 0.4
      prompt is ['vorne links:'];
field vr type is (schaf, wolf, wand, leer)
      weight is 0.4
      prompt is ['vorne rechts:'];
field hl type is (schaf, wolf, wand, leer)
      weight is 0.2
      prompt is ['hinten links:'];
field hr type is (schaf, wolf, wand, leer)
      weight is 0.2
      prompt is ['hinten rechts:'];
field vorne type is (schaf, wolf, wand, leer)
      weight is 0.3
      prompt is ['direkt vor:'];
field nl type is (schaf, wolf, wand, leer)
      weight is 0.3
```

```
            prompt is ['neben links:'];
      field nr type is (schaf, wolf, wand, leer)
            weight is 0.3
            prompt is ['neben rechts:'];
```

**Abgeleitete Eigenschaften**

Diese Eigenschaften werden durch das Programm aus den eingegebenen Werten abgeleitet. Dadurch wird einfach der Aufwand beim Eingeben einer neuen Situation reduziert. Theoretisch hat die Unterscheidung keine Bedeutung. Solche abgeleiteten Eigenschaften haben immer das Gewicht 1 (d.h. die Gewichte der direkt eingegebenen Eigenschaften müssen relativ dazu gewählt werden, damit eine vernünftige Gewichtung entsteht).

```
      repair rule wolf_nah is
            when  vl is wolf or vr is wolf
            then  change wolf_d to direkt_vor;
            end;

      repair rule wolf_fern is
            when  not (vl is wolf or vr is wolf)
            then  change wolf_d to weiter_weg;
            end;
```

**Abstraktionshierarchie der Werte**

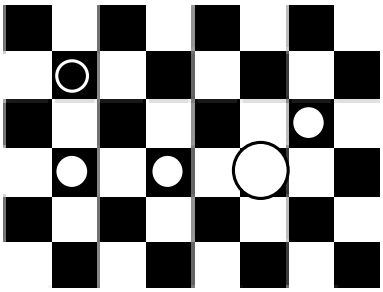
Auf diesem Weg ist es möglich, Werte als ähnlich zu definieren. Wenn eine Eigenschaft bei zwei Fällen nicht identische, sondern nur ähnliche Werte aufweist (z.B. nicht Schaf-Schaf sondern Schaf-Wolf) sinkt das Gewicht dieser Eigenschaft auf 75% des vollen Gewichts.

```
      abstraction figur is (schaf, wolf);
      abstraction hindernis is (figur, wand);
```

**Einige Fälle**

Eine Standardsituation, die Schafe sind nach Plan vorgerückt, S2 (Numerierung von rechts nach links) sollte ziehen und der Wolf blockiert dies nicht:

```
case instance sicher_links_wolf_unsichtbar is
      sn = s2;
      vl = leer;
      vr = schaf;
      hl = leer;
      hr = leer;
      vorne = leer;
      nr = leer;
      nl = schaf;
      wolf_d = weiter_weg;
```



```
solution is
      schritt = 'links';
      nr_neu = schaf;
      hr_neu = leer;
      hl_neu = schaf;

local repair rule definition is
      repair rule links_besetzt is
            when not (vl is leer)
            then
                  reselect;
            end;
end;
```

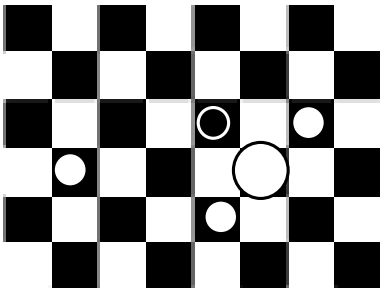
Obwohl die Felder vl und vr ein höheres Gewicht haben, als die anderen Felder, kann es durchaus vorkommen, dass dieser Fall auch dann am besten passt, wenn als einzige

Abweichund auf vl ein Schaf steht. Die "local repair rule" dient als Sicherheitskontrolle und führt dazu, dass der Fall unter diesen Umständen verworfen und ein anderer gesucht wird. (CASPIEN, das Programm, das mit CASL arbeiten kann, kann nicht mehrere Fälle gleichzeitig berücksichtigen, sonst könnte man anstelle der "lokal repair rule" auch stark negative Fälle formulieren).

Der zweite Fall ist eine Situation, in welcher der Wolf direkt vor s2 steht. Das Schaf darf sich in diesem Fall nicht bewegen. Dasselbe gilt immer, wenn der Wolf direkt vor dem Schaf ist.

```
case instance wolf_vor_der_nase is
      sn = s2;
      vr = schaf;
      vl = wolf;
      hl = schaf;
      hr = leer;
      vorne = leer;
      nr = leer;
      nl = leer;
      wolf_d = direkt_vor;

solution is
      schritt = 'stehen bleiben';
end;
```



Entscheidend ist in diesem Fall eigentlich nur, dass der Wolf direkt_vor ist. Diese Eigenschaft hat auch gegenüber allen anderen Eigenschaften ein sehr hohes Gewicht (1), so dass der Fall häufig zum Zuge kommt.

# 5 Überlegungen zu einer komplexeren Repräsentation

Es ist nicht zwingend, "Fall" mit "Brettposition" zu identifizieren. Eine Situation könnte auch mehrere Brettpositionen und die dazwischen liegenden Aktionen umfassen. Dann stellt sich allerdings die Frage, wie eine Situation abgegrenzt wird, wann sie zu Ende ist.

## 5.1 Etappenziele im Spielablauf

Aus der eigenen Introspektion scheint es, das erinnerte Situationen immer mit dem Erreichen eines Etappenziels, mit einer gewissen Antiklimax zu ende gehen, wenn sich die während der laufenden Situation herrschende Spannung und Hektik löst. Auch im Spielverlauf lassen sich solche Etappenziele finden, die Situationsgrenzen bezeichnen könnten.

**Erreichen einer Standardposition**: Die Schafe sind in Kontrolle, wenn sie als geschlossene Kette vorrücken können. Diese Bewegung kann als eine Abfolge von acht Standardpositionen beschreiben werden, die sich zyklisch wiederholen. Kann eine dieser Positionen erreicht werden, können die Schafe in Ruhe die nächste Aktion des Wolfes abwarten. All diesen Positionen gemeinsam ist, dass die Schafe eine geschlossene Kette quer über das Spielfeld bilden.

**Verteidigungsstellung einnehmen**: Blockiert der Wolf im richtigen Moment ein bestimmtes Feld, kann das Standardvorgehen nicht eingehalten werden. Es muss ein anderes Schaf ziehen, als dasjenige, das im Standardablauf an der Reihe wäre. Das führt unweigerlich dazu, dass sich eine Lücke geöffnet. Geschieht dies am richtigen Ort, erreichen die Schafe eine Art optimale Verteidigungsstellung, die es ihnen erlaubt, die Lücke rechtzeitig wieder zu schliessen. Je nachdem in welcher Standardposition eine Blockade auftritt, sieht die korrekte Verteidigungsposition etwas anders aus. Allen gemeinsam ist aber, dass eine Art Zange entsteht, die den Wolf von aussen umfasst.

**Abwenden eines Umgehungsversuches**: Wenn immer sich eine Lücke auf den Seiten auftut, wird der Wolf versuchen, die Schafe auf der Seite der Lücke zu umgehen. Die Abwehr eines solchen Umgehungsversuches ist geglückt, wenn es den Schafen gelingt, sich seitlich-vorwärts so zu verschieben, dass sich die Lücke schliesst ohne dass dadurch eine neue, nicht zu schliessende Lücke entsteht. All diesen Positionen gemeinsam ist, dass eine mehr oder weniger lange Reihe gestaffelt stehender Schafe den Rand berührt, auf dessen Seite der Wolf sich befindet.

**Schliessen einer zentralen Lücke**: Manchmal entstehen zentrale Lücken. Diese wird der Wolf sofort angreifen und sie müssen sofort geschlossen werden. Wenn es gelingt, besteht typischerweise um den Wolf herum eine kleine Zange aus drei Schafe, die ihn zwingt, wieder rückwärts zu gehen.

**Einkreisen des Wolfes**: In der Endphase wird es möglich, den Wolf nicht nur vor sich herzutreiben, sondern in gezielt einzukreisen. Wenn es gelingt, kann sich der Wolf nicht mehr bewegen und das Ziel ist erreicht.

**Durchbruch des Wolfes**: Drohender negativer Abschluss jeder Situation ist der Durchbruch des Wolfes und damit der Verlust des Spiels.

Ich denke, das sind Konzepte, die man bei erfahrenen Schafen erwarten kann. Bei mir haben sie sich ganz klar über die Zeit durch die fortwährende Beschäftigung mit dem Spiel herausgebildet.

### 5.2 Repräsentation der komplexen Episoden

Als einfachste Variante kann die komplexe Episode als zwei oder mehr aufeinanderfolgende Positionen repräsentiert werden. Zu jeder Position (ausser zur letzten) gehört die Aktion, die durchgeführt wurde.

### 5.3 Bewertung der komplexen Episode

Die Bewertung dürfte relativ einfach sein. Sie ist stark negativ, wenn der Wolf durchbrechen kann, stark positiv, wenn die Schafe siegen und schwächer positiv in alle anderen Fällen.

## 6 Vorwissen neuer Schafe

Neue Schafe sollen nicht als unbeschriebenes Blatt ins Spiel kommen, sondern Vorwissen aus einem anderen Kontext mitbringen. Dieser Kontext sollte eine gewisse Ähnlichkeit mit dem Spiel haben, so dass die Schafe einige brauchbare Konzepte mitbringen.

### 6.1 Konzepte, die neue Schafe mitbringen könnten

- Sich auf einem Spielbrett bewegen (rechts, links, vorwärts)
- Wahrnehmung der näheren Umgebung (vorne, hinten, leer, besetzt)
- Direkt erreichbare Felder
- Spielfeldrand als feste Schranke
- Besetzte Felder können nicht betreten werden
- Verbotene Felder

### 6.2 Ein Spiel als Vorerfahrung: Paarlaufen

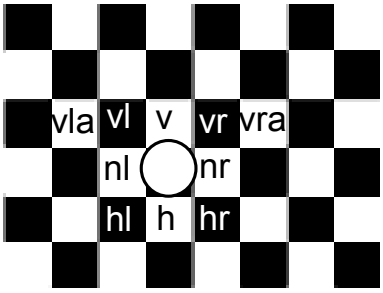
Kleine Schafe lieben es, zu zweit über ein Spielfeld zu streifen. Sie haben daraus ein einfaches Spiel gemacht, das mit folgenden Regeln auskommt:

- Die zwei Partner müssen sich immer in zwei benachbarten Feldern befinden
- Der Partner muss sich immer auf derselben Seite befinden (links oder rechts)

- Es darf immer nur eines der beiden gleichzeitig sich bewegen
- Sie dürfen den Rand des Spielfelds nicht überschreiten
- Felder, auf denen schon ein anderes Schaf steht (Partner oder anderes Paar) dürfen nicht betreten werden

## 6.3 Eine einfache Repräsentation

Die Repräsentation kann ganz ähnlich sein, wie die der erfahrenen Spieler.




- Vorne
- Vorne links
- Vorne rechts
- Vorne links aussen
- Vorne rechst aussen
- Neben links
- Neben rechts
- Hinten
- Hinten links
- Hinten rechts

(mögliche Werte jeweils: schaf, wand oder leer)

- Schritt (was tut das Schaf: links, rechts, stehenbleiben)

Vorne, Hinten, Hinten links und Hinten rechts sind eigentlich überflüssig, erleichtern aber sicher den Transfer auf das Wolfsspiel.

'Vorne links aussen' und 'vorne rechts aussen' sind notwendig, damit sich das Schaf mit seinem Partner nicht in eine Situation manövriert, in der sie beide bewegungsunfähig vor einer Wand stehen. Sie sind allerdings nur bedingt nötig, denn im Prinzip könnte das Schaf dieselbe Information im Austausch mit dem Partner erfahren.

## 6.4 Bewertung der Fälle

Die Bewertung ist einfach. Sie ist positiv, wenn die Aktion in eine erlaubte Situation führt, negativ, wenn eine Regel verletzt wird.

## 6.5 Ein Implementationsbeispiel mit nur positiven Fällen

In CASL sieht eine einfach Darstellung des minimalen Wissens eines erfahrenen Paarläufers etwa wie folgt aus. Alle Fälle sind positiv formuliert, eine Bewertung muss deshalb nicht explizit angegeben werden.

Die wirklich relevanten Felder als Eingabegrössen:

```
case definition is
      field vl type is (schaf, wand, leer)
            prompt is ['vorne links:'];
      field vr type is (schaf, wand, leer)
            prompt is ['vorne rechts:'];
      field vla type is (schaf, wand, leer)
            prompt is ['vorne links aussen:'];
      field vra type is (schaf, wand, leer)
            prompt is ['vorne rechts aussen:'];
      field nl type is (schaf, wand, leer, partner)
            prompt is ['neben links:'];
      field nr type is (schaf, wand, leer, partner)
            prompt is ['neben rechts:'];
end;
```

Dieselbe Abstraktionshierarchie wie bei den "ausgewachsenen" Schafen:

```
modification definition is
      abstraction figur is (schaf, wolf);
      abstraction hindernis is (figur, wand);
end;
```

Eine Vorverarbeitung, welche die Aufmerksamkeit auf die Seite lenkt, auf welcher der Partner steht. Die Felder auf jener Seite bekommen ein doppeltes Gewicht.

```
preprocess rule definition is

      repair rule links is
            when
                  nl is partner
            then
                  pr (['Partner links']);
                  change weight of vl to 2;
                        change weight of vla to 2;
            end;

      repair rule rechts is
            when
                  nr is partner
            then
                  pr (['Partner rechst']);
                  change weight of vr to 2;
                        change weight of vra to 2;
            end;

end;
```

Der Fall, indem das Schaf im Uhrzeigersinn einen Schritt um seinen rechts stehenden Partner macht (für die "lokal repair rule" siehe bei den "erwachsenen" Schafen):

```
case instance kann_rechts is
      vl = leer;
      vr = leer;
      vla = leer;
      vra = leer;
      nr = partner;
      nl = leer;
solution is
      schritt = 'rechts';

local repair rule definition is
      repair rule rechts_besetzt is
            when not (vr is leer)
            then reselect;
            end;
end;
```

Die identische Situation für eine Drehung im Gegenuhrzeigersinn:

```
case instance kann_links is
      vr = leer;
      vl = leer;
      vla = leer;
      vra = leer;
```

```
      nr = leer;
      nl = partner;
solution is
      schritt = 'links';

local repair rule definition is
      repair rule links_besetzt is
            when not (vl is leer)
            then reselect;
            end;
end;
```

Wenn für eine Drehung nach rechts ein Schaf im Wege steht:

```
case instance  rechts_geht_nicht is
      vl = leer;
      vr = wand;
      vla = leer;
      vra = leer;
      nr = partner;
      nl = leer;
solution is
      schritt = 'stehen bleiben';
end;
```

Dito für links:

```
case instance links_geht_nicht is
      vl = wand;
      vr = leer;
      vla = leer;
      vra = leer;
      nr = leer;
      nl = partner;
solution is
      schritt = 'stehen bleiben';
end;
```

Wenn eine Drehung nach rechts das Paar gegen die Wand führen würde:

```
case instance rechts_gegen_wand is
      vl = leer;
      vr = leer;
      vla = leer;
      vra = wand;
      nl = leer;
      nr = partner;
solution is
      schritt = 'stehen bleiben';
end;
```

Dito nach links:

```
case instance links_gegen_wand is
      vl = leer;
      vr = leer;
      vla = wand;
      vra = leer;
      nl = partner;
      nr = leer;
```

```
solution is
      schritt = 'stehen bleiben';
end;
```

# 7 Die Struktur des Vorwissens

Damit das Vorwissen beim Erlernen des Spiels über Strukturvergleiche genutzt werden kann, dürfen die Fälle nicht einfach nur eine Aneinanderreihung von Eigenschaften und Werten sein, sondern müssen auch eine Struktur aufweisen.

Im Vorwissen lassen sich verschiedene Aspekte unterschieden. Die auf verschiedene Art wirksam werden können.

## 7.1 Sich auf einem Schachbrett bewegen

Von erfahrenen Paarläufern kann man erwarten, dass sie sich auf einem Schachbrett bewegen können, dass sie wissen, was sie tun müssen, um ein bestimmtes, direkt angrenzendes Feld zu erreichen.

Dieses Wissen kann einmal als sensomotorisches Wissen vorausgesetzt werden. Am einfachsten so, dass jedes Schaf, wenn es ein bestimmtes Feld zum Ziel macht, in der Lage ist, die geeignete Aktion zu wählen, auszuführen und überprüfen, ob es geklappt hat.

Das Wissen darüber, wann das entsprechende sensomotorische Programm abzurufen ist, kann man sich als situativ-episodische Erinnerungen an geglückte Bewegungen vorstellen. In der von ACME (vgl. 2.2) verwendeten Notation könnten Erinnerungen an entsprechende Situationen etwa wie folgt aussehen:

```
(make_struc 'bewegung_gelückt
            'situation
            '(beteiligte
                 ((schaf (obj_schaf) p1)
                  (selbst (obj_schaf) p2)))
            '(position_vor
                ((feld (obj_ort1) p3)
                 (feld (obj_ort2) p4)
                 (auf (obj_schaf, obj_ort1) p5)
                 (vorne_rechts (obj_ort2, obj_ort1, obj_schaf) p6)))
            '(aktion
                ((bewegen_nach* (obj_ort1, obj_ort2, obj_schaf) p7)))
            '(position_nach
                ((auf (obj_schaf, obj_ort2) p8)))
            '(bewertung
                ((bewertung (10, p7) p9)))
)
```

- "bewegung_geglückt" ist ein nur technisch notwendiger Kurzname für die Situationsbeschreibung
- "p1" etc. sind technisch notwendige Kurznamen für die einzelnen Propositionen, die es erlauben, in einer Proposition auf eine andere zu verweisen (vgl. den Verweis auf p7 in p9).
- "beteiligte", "position_vor" etc. strukturieren die ganze Beschreibung in sinnvolle Abschnitte
- obj_xy sind Referenzen auf bestimmte, konkrete Dinge, die in der Situation präsent waren.
- "schaf", "selbst", "feld" sind Eigenschaften dieser Dinge
- "auf(x, y)" bedeutet, dass x auf dem Feld y steht.
- "vorne_rechts(y, x, z)" bedeutet, dass aus der Perspektive von Schaf z sich das Feld y vorne rechts neben dem Feld x befindet.

- "bewegen_nach*(y,x,z)" bedeutet, dass an dieser Stelle das entsprechende sensomotorische Programm aufgerufen wurde mit dem Ziel, Schaf z von x nach y zu bewegen.
- "bewertung(x,y)" bedeutet, dass die Situation auf einer Skala von x=+10 bis x=-10 als mehr oder weniger vorbildlich bewertet wird. Y ist der Grund dafür.

## 7.2 Nach Spielregeln spielen

Erfahrene Paarläufer wissen, was es bedeutet, sich nicht einfach frei auf einem Schachbrett zu bewegen, sondern Spielregeln zu beachten.

### *7.2.1 Geeignete Orientierung*

Die Vielfalt der wahrgenommen Situationen reduziert sich, wenn das Schaf seine Wahrnehmung geeignet orientiert. Beim Paarlaufen ist dies der Fall, wenn es sich nach jedem Schritt (selbst oder durch den Partner ausgeführt) sofort so dreht, dass der Partner nebenan (auf der vorgeschrieben Seite) steht.

Dieses Wissen kann ebenfalls einmal als sensomotorisches Wissen mit entsprechendem Automatismus vorausgesetzt werden. Für erfahrene Paarläufer kann man annehmen, dass dies direkt mit dem eigentlichen Schritt verschmolzen ist, d.h. dass die entsprechende Drehung als Teil des Schrittes erlebt wird.

Für das Wolfsspiel muss diese Verschmelzung erstens einmal aufgebrochen werden. Anlass dazu dürfte sein, dass kein "Partner" definiert ist und somit auch unklar ist, wie die Ausrichtung erfolgen kann. Damit ein Transfer möglich ist, muss es dann aber möglich sein, der automatischen Ausrichtung eine neue Richtung zu geben. Beim Wolfsspiel ist die sinnvollste Ausrichtung der konstante Blick direkt nach vorne.

### *7.2.2 Spielregeln befolgen*

Spielregeln werden befolgt, d.h. erfahrene Schafe führen nur Aktionen aus, die erlaubt sind. Dies ist einmal natürlich als situativ-episodisches Wissen verankert, indem dort entsprechende Fälle positiv bzw. negativ gekennzeichnet sind.

Ein Transfer ist z.B. möglich, wenn es negative Fälle gibt, bei denen die Begründung in deklarativer Form vorliegt. Beziehungen, die nützlich sein könnten, sind:

- verboten(feld1, schaf1): die Regeln verbieten dem schaf1 das Betreten des Feldes in der aktuellen Situation
- grund(verbot1, regel1): der Grund für das verbot ist eine bestimmte Regel
- regel(feld2, logische Kombination von Eigenschaften von Feldern, "verboten" oder "erlaubt")

Konkrete Beispiele:

**Regeln**

- Regel(x , wand(x), verboten, r1)
  *Die Regel bezieht sich auf Feld x und sagt aus, dass das Betreten dieses Feldes verboten ist, wenn sich dort die Wand, d.h. der Spielfeldrand, befindet.*
- Regel(x, auf(y, x) and (not(selbst (y))), verboten, r2)
- Regel(x, not(rechts (z, x) and auf(y,z) and partner(y, v, rechts) and selbst(v), verboten, r3)
  *Eine analoge Regel wird für partner(y, v, links) benötigt.*
- Regel(*, bewegen_nach(*, *, s1) and bewgen_nach(*, *, s2) and partner(s1, s2, *), verboten, r4)
  *Es dürfen sich nicht beide Partner gleichzeitig bewegen. * steht für beliebige Werte..*

**Situationen**

In der von ACME (vgl. 2.2) verwendeten Notation könnten Erinnerungen an entsprechende Situationen etwa wie folgt aussehen:

- **Verletzung von Regel r1**: Das Schaf bewegt sich auf ein Feld (obj_ort3), das von der Wand "belegt" ist.

```
(make_struc 'regel_1_verletzt
              'situation
           '(beteiligte
                 ((schaf (obj_schaf) p1)
                  (selbst (obj_schaf) p2)
                  (schaf (obj_schaf2) p3)
                  (partner (obj_schaf2, obj_schaf, rechts) p4)))
              '(position_vor
                ((feld (obj_ort1) p5)
                 (feld (obj_ort2) p6)
                 (feld (obj_ort3) p7)
                 (auf (obj_schaf, obj_ort1) p8)
                 (auf(obj_scha2, obj_ort2) p9)
                 (rechts(obj_ort2, obj_ort1, obj_schaf) p10)
                 (vorne_rechts (obj_ort3, obj_ort1, obj_schaf) p11)
                 (wand (obj_ort3) p12)))
              '(aktion
                ((bewegen_nach* (obj_ort1, obj_ort3, obj_schaf) p13)))
              '(position_nach
                ((auf (obj_schaf, obj_ort3) p14)
                 (auf (obj_schaf1, obj_ort2) p15)
                 (rechts (obj_ort2, obj_ort3, obj_schaf) p16)))
              '(bewertung
                ((verboten (obj_ort3, obj_schaf) p17)
                 (grund (p17, r1) p18)
                 (bewertung (-10, p17) p19)))
)
```

- "wand(x)" ist eine neue Eigenschaft für ein Feld und bedeutet, dass dieses "Feld" ausserhalb der Spielfeldgrenze liegt.
- "partnet(x,y,z)" bedeutet, dass x und y zusammen ein Paar bilden, und dass sich y immer z=rechts oder z=links von x aufhalten muss.
- "rechts" und "vorne_links" (s. das dritte Beispiel) sind analog "vorne_rechts" zu verstehen.

- **Verletzung von Regel r2**: Das Schaf betritt ein besetztes Feld (obj_ort3).

```
(make_struc 'regel_2_verletzt
              'situation
           '(beteiligte
                 ((schaf (obj_schaf) p1)
                  (selbst (obj_schaf) p2)
                  (schaf (obj_schaf2) p3)
                  (partner (obj_schaf2, obj_schaf, rechts) p4)
                  (schaf (obj_schaf3) p41)))
              '(position_vor
                ((feld (obj_ort1) p5)
                 (feld (obj_ort2) p6)
                 (feld (obj_ort3) p7)
                 (auf (obj_schaf, obj_ort1) p8)
                 (auf(obj_scha2, obj_ort2) p9)
                 (rechts(obj_ort2, obj_ort1, obj_schaf) p10)
                 (vorne_rechts (obj_ort3, obj_ort1, obj_schaf) p11)
                 (auf (obj_schaf3, obj_ort3) p12)))
              '(aktion
                ((bewegen_nach* (obj_ort1, obj_ort3,obj_schaf) p13)))
              '(position_nach
                ((auf (obj_schaf, obj_ort3) p14)
                 (auf (obj_schaf1, obj_ort2) p15)
               (auf (obj_schaf3, obj_ort3) p151)
                 (rechts (obj_ort2, obj_ort3, obj_schaf) p16)))
              '(bewertung
                ((verboten (obj_ort3, obj_schaf) p17)
                 (grund (p17, r2) p18)
                 (bewertung (-10, p17) p19)))
)
```

- **Verletzung von Regel r3**: Das Schaf verliert den Kontakt zum Partner..

```
(make_struc 'regel_3_verletzt
              'situation
           '(beteiligte
                 ((schaf (obj_schaf) p1)
                  (selbst (obj_schaf) p2)
                  (schaf (obj_schaf2) p3)
                  (partner (obj_schaf2, obj_schaf, rechts) p4)))
              '(position_vor
                ((feld (obj_ort1) p5)
                 (feld (obj_ort2) p6)
                 (feld (obj_ort3) p7)
                 (auf (obj_schaf, obj_ort1) p8)
                 (auf(obj_scha2, obj_ort2) p9)
                 (rechts(obj_ort2, obj_ort1, obj_schaf) p10)
                 (vorne (obj_ort3, obj_ort1, obj_schaf) p11)))
              '(aktion
                ((bewegen_nach* (obj_ort1, obj_ort3, obj_schaf) p13)))
              '(position_nach
                ((auf (obj_schaf, obj_ort3) p14)
                 (auf (obj_schaf1, obj_ort2) p15)
                 (vorne_links (obj_ort2, obj_ort3, obj_schaf) p16)))
              '(bewertung
                ((verboten (obj_ort3, obj_schaf) p17)
                 (grund (p17, r3) p18)
                 (bewertung (-10, p17) p19)))
)
```

## 7.3 Eine neue Regel anwenden

Damit diese Situationen überhaupt so erinnert werden können, musste einmal eine solche Situation durch Anwendung der deklarativ gegeben Regel bewertet werden. Das Wissen, wie ein solche Bewertung abläuft, kann als prozedurales Wissen vorausgesetzt werden. Die Auslösesituation, d.h. das Wissen, dass dieser Prozess gestartet werden muss, kann auch hier in Form von situativ-episodischen Erinnerungen angenommen werden.

## 7.4 Mit Mitspielern kooperieren

Da sich immer nur ein Schaf auf einmal bewegen darf, müssen sich die beiden Partner absprechen, wer den nächsten Schritt macht. Auch hier kann man prozedurales Wissen zur Steuerung des Aushandlungsprozess situativ-episodische Erinnerungen als Auslösen annehmen.

### *7.4.1 Auslösung*

```
(make_struc 'verhandeln_vor_handeln
              'situation
              '(beteiligte
                 ((schaf (obj_schaf) p1)
                  (selbst (obj_schaf) p2)
                  (schaf (obj_schaf2) p3)
                  (partner (obj_schaf2, obj_schaf, rechts) p4)))
              '(position_vor
                ((feld (obj_ort1) p5)
                 (feld (obj_ort2) p6)
                 (feld (obj_ort3) p7)
                 (auf (obj_schaf, obj_ort1) p8)
                 (auf(obj_scha2, obj_ort2) p9)
                 (rechts(obj_ort2, obj_ort1, obj_schaf) p10)
                 (vorne_rechts (obj_ort3, obj_ort1, obj_schaf) p11)
                 (vorschlag (obj_ort1, obj_ort3, obj_schaf,
                             obj_schaf) p12)))
              '(aktion
                ((verhandeln* (obj_schaf, obj_schaf2, p12) p121)
                 (bewegen_nach* (obj_ort1, obj_ort3, obj_schaf) p13)
                 (sequenz (p121, p13) p131)))
              '(position_nach
                ((auf (obj_schaf, obj_ort3) p14)
                 (auf (obj_schaf1, obj_ort2) p15)
                 (rechts (obj_ort2, obj_ort3, obj_schaf) p16)))
              '(bewertung
                ((bewertung (10, p131) p17)))
)
```

- "vorschlag(x,y,z, v)" steht dafür, dass v in dieser Situation vorschlagen würde, dass z von x nach y geht.
- "verhandeln*(x, y, z, u, v, w)" heisst, dass an dieser Stelle das prozedurale Programm zur Abstimmung der Aktionen aufgerufen wurde. X verhandelt mit y und bringt als erstes Vorschlag z ein. Das Resultat ist, dass u sich von v nach w bewegen soll.
- "sequenz(x, y)" dokumentiert, dass die beiden "Programme" x und y in dieser Reihenfolge ausgeführt wurden.

### *7.4.2 Ablauf*

Formal lässt sich der Ablauf von Verhandlungen als Produktionssystem beschreiben. Ein Ausarbeitung der benötigten Regeln würde den Rahmen dieser Überlegungen sprengen und muss später noch erfolgen.

## 7.5 Perspektivenwechsel

In verschieden Momenten ist es nützlich, wenn das Schaf in der Lage ist, aus der eigenen Perspektive "umzurechnen", wie die Situation für den Partner aussieht. Im Prinzip lässt sich auch das als situativ-episodisches Wissen festhalten, bzw. als Erinnerung an eine Situation, in der das Schaf sich einmal intensiv mit dem Partner über ihre unterschiedlichen Perspektiven unterhalten haben. Packt man alles in eine einzige Situation, dann ergibt das z.B.

```
(make_struc 'verhandeln_vor_handeln
              'situation
              '(beteiligte
                 ((schaf (obj_schaf) p1)
                  (selbst (obj_schaf) p2)
                  (schaf (obj_schaf2) p3)
                  (partner (obj_schaf2, obj_schaf, rechts) p4)))
              '(position_vor
                ((feld (obj_ort1) p5)
                 (feld (obj_ort2) p6)
                 (feld (obj_ort3) p7)
                 (feld (obj_ort4) p8)
                 (feld (obj_ort5) p9)
                 (feld (obj_ort6) p10)
                 (feld (obj_ort7) p11)
                 (feld (obj_ort8) p12)
                 (auf (obj_schaf, obj_ort5) p13)
                 (auf(obj_scha2, obj_ort6) p14)
                 (rechts(obj_ort6, obj_ort5, obj_schaf) p15)
                 (vorne_links (obj_ort1, obj_ort5, obj_schaf) p16)
                 (vorne (obj_ort2, obj_ort5, obj_schaf) p17)
                 (vorne_rechts (obj_ort3, obj_ort5, obj_schaf) p18)
                 (vorne_rechts_aussen (obj_ort4, obj_ort5, obj_schaf) p19)
                 (hinten (obj_ort7, obj_ort5, obj_schaf) p20)
                 (hinten_rechts (obj_ort8, obj_ort5, obj_schaf) p21)
                 (perspektiven_vergleich* (obj_schaf, obj_schaf2) p22)
                 (links(obj_ort5, obj_ort6, obj_schaf2) p23)
                 (vorne_links (obj_ort2, obj_ort6, obj_schaf2) p24)
                 (vorne (obj_ort3, obj_ort6, obj_schaf2) p25)
                 (vorne_rechts (obj_ort4, obj_ort6, obj_schaf2) p26)
                 (hinten (obj_ort8, obj_ort6, obj_schaf2) p27)
                 (hinten_links (obj_ort7, obj_ort6, obj_schaf2) p28)))
              '(bewertung
                ((bewertung (10, p28) p22)))
)
```

## 7.6 Mehrere Schritte vorausplanen

Es gibt eine Konstellation, in der eine zwar legale Bewegung in eine Situation führt, in der beide Partner direkt vor der Wand stehen und so nicht weiter können (vgl. 6.2 und 6.5). Diese Erfahrung könnte sich im Wissen niederschlagen, dass es sich lohnt, zumindest einen Schritt vorauszudenken – und auch wie man das macht.

Im Prinzip haben die Schafe ihr Spiel "verloren", wenn sie sich in eine solche Situation manöverieren. Dies kann einmal durch die Einführung einer weiteren Regel festgehalten werden.

- Regel(*, selbst(x) and partner(y, x, rechts) and auf(x, a) and auf(y, c) and vorne_rechts(b, a, x) and verboten(b, x) and vorne_links(d, c, y) and verboten(d, y), verloren, r5)

Diese Regel ist sehr spezifisch für das Paarlaufen, ob sie eine gute Ausgangslage bildet, um das Wolfsspiel zu erlernen, muss sich noch zeigen.

Damit lässt sich nun eine Situation beschreiben, in der die Schafe Gefahr laufen, zu verlieren.

```
(make_struc 'würde_verlieren
              'situation
           '(beteiligte
                 ((schaf (obj_schaf) p1)
                  (selbst (obj_schaf) p2)
                  (schaf (obj_schaf2) p3)
                  (partner (obj_schaf2, obj_schaf, rechts) p4)))
              '(position_vor
                ((feld (obj_ort1) p5)
                 (feld (obj_ort2) p6)
                 (feld (obj_ort3) p7)
                 (feld (obj_ort4) p71)
                 (auf (obj_schaf, obj_ort1) p8)
                 (auf(obj_schaf2, obj_ort2) p9)
                 (wand (obj_ort4) p91)
                 (rechts(obj_ort2, obj_ort1, obj_schaf) p10)
                 (vorne_rechts (obj_ort3, obj_ort1, obj_schaf) p11)
                 (vorne_rechts_rechts_aussen(obj_ort4, obj_ort1,
                                             obj_schaf) p111)))
              '(aktion
                ((bewegen_nach* (obj_ort1, obj_ort3, obj_schaf) p13)))
              '(position_nach
                ((auf (obj_schaf, obj_ort3) p14)
                 (auf (obj_schaf1, obj_ort2) p15)
                 (rechts (obj_ort2, obj_ort3, obj_schaf) p16)
                 (vorne_links (obj_ort4, obj_ort3, obj_schaf2) p161)
                 (vorne_rechts (obj_ort5, obj_ort3, obj_schaf) p162)
                 (wand (obj_ort5) p163)))
              '(bewertung
                ((verloren (obj_schaf) p17)
                 (grund (p183, r5) p18)
                 (bewertung (-10, p17) p19)))
)
```

Damit ist ein solches Schaf natürlich noch nicht in der Lage, mehrere Schritt voraus zu planen. Ob dies nötig sein wird und ob sich das aus diesen Anfängen entwickeln kann, bleibt abzuwarten.