Agenten in der Wissenschaft: Der RoboCup

Dr. Klaus Dorer, Director Technology Research living systems AG
Christian Dannegger, CTO living systems AG

Abstract

Zur Demonstration der Leistungsfähigkeit von Agententechnologie und zum Vergleich verschiedener Forschungsansätze wurde Mitte der 90er Jahre eine Standardumgebung für Agenten geschaffen: Fußball. Wichtige Eigenschaften von Agenten wie z.B. Autonomie, Kollaboration mit anderen Agenten und Flexibilität bei schnellen und unvorhergesehenen Änderungen in der Umgebung können in dieser Umgebung optimal untersucht und demonstriert werden. Seit 1997 finden jährlich Weltmeisterschaften im Roboter- und Agentenfußball statt, bei denen sich Agentenmannschaften aus mehr als 20 Ländern messen. Mit Hilfe der auf CD mitgelieferten Java-Klassen und -Interfaces können jetzt auch Java-Agenten Fußball spielen. Der folgende Artikel soll den Einstieg in agentenorientierte Softwareentwicklung anhand der besonders spannenden Domäne Fußball erleichtern.

RoboCup

Seit Schachweltmeister Kasparov von einem Computer besiegt wurde eignet sich Schach nur noch bedingt als ‚Standarddomäne' in der künstlichen Intelligenz Forschung. Daher hielt man Ausschau nach einer neuen, schwierigeren Domäne und fand sie im Fußball. Das klingt zunächst Paradox, da ja schon kleine Kinder Fußball spielen können, man Schach aber in der Regel erst wesentlich später lernt. Man muss dabei aber bedenken, dass viele der Intelligenzleistungen beim Fußball dem Menschen bereits angeboren sind und deswegen nicht als solche empfunden werden. Konkret ist die Schwierigkeit der Fußballdomäne durch folgende Eigenschaften gegeben. Sie ist kontinuierlich: bei Schach gibt es genau 64 Felder, im Fußball ist die Zahl der Positionen praktisch unbegrenzt. Sie ist nicht-deterministisch: ein Zug beim Schach wird immer das gewünschte Ergebnis liefern, im Fußball wird kein Schuss wie der andere gelingen. Die Fußballdomäne ist dynamisch: beim Schach wird die Umgebung nur durch den Spieler und einen Gegenspieler verändert, beim Fußball sind es neben 10 Mitspielern und 11 Gegnern unter anderem auch noch Umwelteinflüsse. Weiterhin ist sie kollaborativ: ein Agent ist nur dann erfolgreich, wenn er mit den anderen Agenten (Mitspielern) zusammenarbeitet. Schließlich ist Fußball auch gegnerisch: die Gegneragenten versuchen aktiv, den Erfolg des Agenten zu vermeiden.

Seit 1997 werden alljährlich Weltmeisterschaften im Roboterfußball ausgetragen [1]. Gespielt wird in 4 Ligen: der Simulationsliga, in der 22 reine Softwareagenten spielen, der Sony legged robots league, in der sich jeweils drei vierbeinige Aibo Roboter von Sony gegenüber stehen, der small sized league, in der jeweils 5 Roboter auf einem Feld von der Größe einer Tischtennisplatte gegeneinander spielen, sowie der middle sized league, bei der jeweils 4 Roboter mit maximal 50 cm Durchmesser antreten. In dieser Zeit stieg sowohl die Zahl der Teilnehmer (siehe Tabelle 1) als auch die Zahl der Zuschauer enorm. Für den RoboCup 2001, der vom 3. bis 10. August in Seattle stattfindet, wird mit weltweit drei Millionen Zuschauern gerechnet.

living systems AG - Tel.: +49 (0)771/8987-0 - Fax: +49 (0)771/8987-100 – www.living-systems.com

	1997	1998	1999	2000	2001
Simulationsliga	33	46	39	42 (62)	42 (81)
Small Size League	4	13	18	23	29
Middle Size League	5	22	20	24	21
Sony Legged Robot League	-	-	4	12	16

Tabelle 1: Anzahl der am RoboCup teilnehmenden Mannschaften. In Klammern steht die für die Vorausscheidung angemeldete Anzahl der Mannschaften.

	1997	1998	1999	2000
Simulationsliga	ATHumboldt Berlin	CMUnited Pittsburgh	CMUnited Pittsburgh	FCPortugal Porto
Small Size League	CMUnited Pittsburgh	CMUnited Pittsburgh	Cornell Big Red Ithaca	Cornell Big Red Ithaca
Middle Size League	Dreamteam Los Angeles und Trakies Osaka	CSFreiburg	Sharif CE Teheran	CSFreiburg
Sony Legged Robot League	-	-	LPR Paris	UNSW Sydney

Tabelle 2: Die Weltmeister.

Simulationsliga

Die Simulationsliga ist die Liga der Softwareagenten. In ihr wird die Umgebung durch ein Serverprogramm simuliert. Es implementiert die physikalischen Eigenschaften der Umgebung, des Balls und der Spieler. Es liefert die Wahrnehmung für die Agenten und führt deren Aktionen aus, indem es die Umgebung entsprechend ändert. Ebenso dient es als Schiedsrichter, um unter anderem auf Tor, Einwurf, Eckball oder Abseits zu entscheiden.

Wahrnehmungen eines Agenten sind beispielsweise die ungefähre Richtung und Entfernung der anderen Spieler, der Tore oder des Balls, falls sich diese in Blickrichtung des Agenten befinden. Ungefähr deshalb, da der Server die Wahrnehmungen und Aktionen eines Spielers verrauscht, um reale Sensoren und Aktoren zu simulieren. Ein Agent kann zwischen 45, 90 und 180 Grad Blickwinkel entscheiden, wobei die Wahrnehmungen dann alle 75, 150 bzw. 300ms beim Spieler eintreffen. Die Wahrnehmung ist immer lokal, d.h. jeder Agent hat seine eigene Sicht von der Welt, abhängig von Position und Blickrichtung. Alle Arten der Wahrnehmung sind im Programmbeispiel 1 aufgeführt.

living systems AG - Tel.: +49 (0)771/8987-0 - Fax: +49 (0)771/8987-100 – www.living-systems.com

Aktionen kann ein Agent alle 100ms ausführen. Die Aktionen beinhalten unter anderem das Drehen des Agenten um die eigene Achse, das Drehen des Kopfs, Fortbewegung vorwärts und rückwärts, den Ball zu schießen oder etwas zu sagen (siehe Programmbeispiel 2). Die Ausdauer der Agenten ist dabei ebenso begrenzt wie die maximale Geschwindigkeit und die Bandbreite der Kommunikation.

Auf der mitgelieferten CD finden Sie bereits die nötigen Klassen, die die Kommunikation mit dem Server sowie das Parsen der Wahrnehmungen übernehmen. Außerdem ist im Paket ein einfacher Beispielagent enthalten, der bereits in der Lage ist, ein Tor zu erzielen. Sie können dadurch sehr einfach Ihre eigenen Fußball-Agenten in Java programmieren, ohne sich um die technischen Details der Kommunikation mit dem Server kümmern zu müssen. Sie müssen lediglich das frei verfügbare Serverprogramm mit Viewer zur Darstellung des Spiels installieren (siehe [2]), den mitgelieferten Beispielagenten starten und nach Belieben erweitern.

living systems AG - Tel.: +49 (0)771/8987-0 - Fax: +49 (0)771/8987-100 – www.living-systems.com

```
/**   Interface aller möglichen Wahrnehmungen eines Agenten.
      Die entsprechenden Methoden werden automatisch beim Erhalt
      einer Wahrnehmung von der mitgelieferten ServerStub-Klasse
      aufgerufen */

public interface ISoccerPerception
{
      /** Wahrnehmung über die ungefähre Richtung und Entfernung
      aller Agenten, Tore, des Balls und Orientierungspunkte im
      Blickfeld (45, 90 oder 180 Grad) des Agenten. Je weiter
      entfernt desto ungenauer. */
      void perceptionSee(String message);

      /** Wird aufgerufen, wenn ein Spieler im Umkreis von 50m, der
      Trainer oder der Schiedsrichter etwas gesagt haben */
      void perceptionHear(String message);

      /** Körperinformation, wie aktuelle Ausdauer, Kopfdrehung,
      usw. eines Agenten. Wird alle 100ms aufgerufen */
      void perceptionSenseBody(String message);

      /** Anmeldebestätigung mit u.a. der Spielernummer */
      void perceptionInit(String message);

      /** Information vor Spielstart über die Parametereinstellung
      des Servers (z.B. maximale Ballgeschwindigkeit) */
      void perceptionServerParam(String message);

      /** Information vor Spielstart über die Parametereinstellung
      des Servers für Agenten (z.B. Gewicht) */
      void perceptionPlayerParam(String message);

      /** Information vor Spielstart über die 7 Spielertypen aus
      denen ausgewählt werden kann */
      void perceptionPlayerType(String message);

      /** Wird aufgerufen, wenn eine Mannschaft einen Agenten
      gegen einen neuen austauscht (max. 3 Auswechslungen) */
      void perceptionChangePlayer(String message);

      /** Information über den Spielstand bei einem Tor oder nach
      der score() Aktion */
      void perceptionScore(String message);

      /** Alle anderen Wahrnehmungen (z.B. Fehlermeldungen) */
      void perceptionMessage(String message);
}
```

Programmbeispiel 1: Wahrnehmungen eines Agenten in der Fußballumgebung. Eine genaue Beschreibung der Parameter findet man in [2].

living systems AG - Tel.: +49 (0)771/8987-0 - Fax: +49 (0)771/8987-100 – www.living-systems.com

```
/** Interface aller möglichen Aktionen eines Agenten */
public interface ISoccerAction
{
      /** positioniert den Agenten in der eigenen Hälfte vor dem
      Beginn einer Halbzeit oder nach einem Tor */
      void move(int x, int y);

      /** dreht den Körper des Agenten ungefähr mit dem angegebenen
      Winkel (-180 bis 180 Grad) um die eigene Achse. Die tatsächl.
      Drehung hängt von der Geschwindikeit des Agenten ab. */
      void turn(int moment);

      /** dreht den Kopf des Agenten um den angegebenen Winkel */
      void turnNeck(int angle);

      /** beschleunigt den Agenten mit der angegebenen Kraft. Werte
      von -100 (rückwärts) bis 100. Der Agent wird dabei müde */
      void dash(int power);

      /** beschleunigt den Ball ungefähr mit der angegebenen Kraft
      (0 - 100) ungefähr in die angegebene Richtung, falls der Ball
      in Reichweite des Agenten (ca. 1m) liegt */
      void kick(int power, int direction);

      /** fängt den Ball (nur Torwart), wenn er in einem Rechteck
      (2m x 1m) in der angegebenen Richtung vom Agenten ist */
      void catchBall(int direction);

      /** Schickt eine Nachricht an andere Agenten im Umkreis von
      50m. Im Mittel alle 2 Sekunden und nur 512 Byte möglich */
      void say(String message);

      /** ändert das Sehen des Agenten. Beim Blickwinkel 45, 90,
      180 Grad sieht der Agent alle 75, 150, 300 ms etwas */
      void changeView(String angle, String quality);

      /** fragt den Schiedsrichter nach dem Spielstand (nötig falls
      Verbindungsprobleme bestanden und ein Tor verpasst wurde) */
      void score();

      /** fragt den Server nach Körperwahrnehmung. Wird ab server
      Version 6 automatisch alle 100ms gesendet */
      void senseBody();

      /** meldet den Agenten beim Server an */
      void connect(String team, String version, boolean goalie);

      /** meldet den Agenten zurück nach der Halbzeit */
      void reconnect(String team, int number);

      /** meldet den Agenten vom Server ab */
      void disconnect();
}
```

Programmbeispiel 2: Aktionen eines Agenten in der Fußballumgebung.

living systems AG - Tel.: +49 (0)771/8987-0 - Fax: +49 (0)771/8987-100 – www.living-systems.com

Fußball–Agenten

Wie bereits in den beiden vorigen Artikeln erläutert, zeichnen sich Agenten unter anderem durch Autonomie, Kollaboration und Mobilität aus. Im folgenden wird beispielhaft beschrieben, wie die Fußball-Agenten von living systems, die natürlich an der RoboCup WM teilnehmen, diese Eigenschaften umsetzen. Mobilität spielt dabei in dieser Umgebung keine Rolle, da sich die Agenten zwar innerhalb des Spielfelds bewegen, aber nicht ihre Umgebung wechseln müssen.

Autonomie

Wie generell bei Agenten besteht das Problem darin, dem Agenten beizubringen, wie er sich aufgrund seiner Wahrnehmungen und seines Vorwissens verhalten soll. In der Simulationsliga müssen die Agenten während der gesamten Spieldauer von 12 Minuten autonom handeln. Dafür existieren verschiedenste Ansätze in der Agenten Forschung, die von den teilnehmenden Mannschaften umgesetzt wurden und im RoboCup miteinander verglichen werden können.

Einige Mannschaften (z.B. Humboldt Berlin, CSRI Sydney) verwenden sogenannte Belief-Desire-Intention (BDI) Architekturen [8], bei denen der Agent neben dem aktuellen Zustand auch Wünsche und Ziele repräsentieren kann. Andere basieren die Handlungskontrolle der Agenten auf Entscheidungsbäume (CMU Pittsburgh, Universität Linköping) oder auf logikbasierte KI-Sprachen wie Prolog (Universität Koblenz). Eine genaue Beschreibung der Mannschaften findet man in [3-6].

Handlungskontrolle wird bei den living systems Agenten mit Hilfe der in dynamischen Umgebungen eingesetzten erweiterten Verhaltensnetzwerke (extended behavior networks: EBN) realisiert [7]. Diese bieten eine Kombination aus proaktiver und reaktiver Handlungskontrolle (siehe Artikel 1). Sie repräsentieren die Ziele des Agenten mit Wichtigkeit und Relevanz, die Verhaltensregeln mit Vorbedingung, Aktionen, Effekten und den benötigten Ressourcen, sowie die Wahrnehmungen. Die Ziele werden mit den Verhaltensregeln und Wahrnehmungen in ein Netzwerk verknüpft, mit dessen Hilfe der Agent seine Handlungsalternativen bewertet. Dabei beeinflussen die Ziele und die Wahrnehmungen das Verhalten gleichermaßen, so dass der Agent sowohl zielgerichtet handeln als auch schnell auf Änderungen in der Umgebung reagieren kann.

Die Aufgabe eines Entwicklers besteht dann aus drei Teilen:

- Das Fachwissen der Agenten muss in Form von Zielen und Regeln definiert und als EBN-Konfigurationsdatei abgelegt werden. In der Fußball-umgebung unterscheiden wir vier Agentenrollen: Torwart, Abwehrspieler, Mittelfeldspieler und Stürmer. Ein Auszug aus einer solchen Konfigurationsdatei eines Abwehrspielers ist in Programmbeispiel 3 dargestellt. Durch die Konfiguration eines Agenten beispielsweise mit dem Torwart-EBN wird dieser Agent dann ein Torwart.
- Für die in der Konfigurationsdatei verwendeten Wahrnehmungen müssen Java-Methoden zur Verfügung gestellt werden. Aufbauend auf den primitiven Wahrnehmungen, die der Server liefert, werden so Wahrnehmungen mit Fachwissen definiert, wie z.B., ob der Ball nahe am gegnerischen Tor ist (Programmbeispiel 4).
- Die in der Konfigurationsdatei definierten Aktionen müssen in Java Methoden umgesetzt werden (Programmbeispiel 5). Welche Methoden aufgerufen werden, entscheidet der Agent mit Hilfe des EBNs.

living systems AG - Tel.: +49 (0)771/8987-0 - Fax: +49 (0)771/8987-100 – www.living-systems.com

Generell bietet dieser Ansatz drei wesentliche Vorteile:

- Durch die klare Trennung von Handlungsentscheidung und Handlungsausführung können basierend auf denselben Wahrnehmungen und Aktionen unterschiedliche Handlungskontrollmechanismen eingesetzt werden. Neben EBNs können so beispielsweise in weniger dynamischen Umgebungen Workflow-Engines die Kontrolle des Agenten übernehmen.
- Die einzelnen Wahrnehmungen und Aktionen sind klar voneinander abgegrenzt und können relativ unabhängig voneinander entwickelt und von verschiedenen Agenten verwendet werden. Im Idealfall kann ein neuer Agent einfach durch Definition einer EBN – oder Workflow-Konfigurationsdatei erstellt werden.
- Die Business-Logik ist extern in den EBN-Konfigurationsdateien (bzw. Workflow-Dateien im Fall einer Workflow-Engine) repräsentiert. Sie kann daher vom Domänenexperten einfach erstellt und verändert werden. Änderungen können dem Agenten zur Laufzeit mitgeteilt werden, so dass dieser sein Verhalten ohne Neustart direkt ändert. Dies ist gleichzeitig eine notwendige Voraussetzung dafür, dass Agenten Wissen austauschen und lernen können.

Wichtig ist, dass EBNs domänenunabhängig in der Lage sind, Business Logik abzubilden. Regeln und Ziele für Fußball können genauso repräsentiert werden wie das Domänenwissen eines Reise-Agenten oder die Entscheidungskompetenz eines Auktions-Agenten.

```
<extendedBehaviorNetwork>
     <goal>
         <condition>ballInGoal</condition>
         <importance>0.9</importance>
         <relevance>ballIsNearOtherGoal</relevance>
     </goal>

     <goal>
         <condition>haveBall</condition>
         <importance>0.6</importance>
     </goal>
     ...

     <competence>
         <condition>ballKickable</condition>
         <condition>nearOpponent</condition>
         <action>kickAwayBall</action>
         <effect>
             <name>teammateHasBall</name>
             <probability>0.5</probability>
         </effect>
         <resource>
             <name>leg</name>
             <amount>1</amount>
         </resource>
     </competence>
     ...
</extendedBehaviorNetwork>
```

Programmbeispiel 3: Ausschnitt aus der ‚Business Logik' eines Abwehrspielers.

living systems AG - Tel.: +49 (0)771/8987-0 - Fax: +49 (0)771/8987-100 – www.living-systems.com

```
public double perceptionBallIsNearOtherGoal()
{
      Ball    ball        = perception.getBall();
      Goal    otherGoal   = perception.getOtherGoal();
      float   distance    = ball.getDistanceTo(otherGoal);

      // der Ball sei nahe am Tor bis 5 m und nicht nahe ab 25 m
      // Abstand vom Tor. Dazwischen wird linear interpoliert.
      return getLinearFuzzyValue(distance, 25.0, 5.0);
}
```

Programmbeispiel 4: ‚High-level' Wahrnehmung mit Domänenwissen.

```
public void actionKickAwayBall()
{
      Position kickTo;

      if (inOwnHalf()) {
          // in der eigenen Hälfte Befreiungsschläge über außen
          if (onLeftSide()) {
              kickTo = perception.getLeftMidlineFlag();
          } else {
              kickTo = perception.getRightMidlineFlag();
          }
      } else {
          // in der gegnerischen Hälfte Richtung Tor schiessen
          kickTo = perception.getOtherGoal();
      }

      // mit voller Kraft in die gewünschte Richtung schiessen
      kick(100, kickTo.getDirection());
}
```

Programmbeispiel 5: ‚High-level' Aktion mit Domänenwissen.

Kollaboration

Es ist klar, dass der Erfolg beim Fußball vom Zusammenspiel aller Spieler abhängt. Besonders deutlich wird dies beim Aufbau einer Abseitsfalle. Nur wenn alle Spieler koordiniert die Defensive verlassen, gelingt es, den Gegner Abseits zu stellen. Bereits das Versagen eines Spielers macht den Gesamterfolg zunichte.

Der Aufbau einer Abseitsfalle basiert bei living systems Agenten auf zwei Elementen. Zum einen kommunizieren die Agenten während des Spiels, indem sie sich ihre Positionen zurufen. Dadurch können die Spieler den Aufbau der Abseitsfalle kontrollieren und koordinieren, ohne den Blick vom Ball abwenden zu müssen. Bleibt ein Spieler zurück, weil er beispielsweise sehr müde ist, brechen auch die anderen Spieler den Aufbau der Abseitsfalle ab, entsprechend ihrer Regeln im Verhaltensnetzwerk. Zum anderen einigen sich die Spieler vor dem Spiel darauf, in welchen Situationen eine Abseitsfalle aufgebaut wird (locker room agreement). Der Start der Abseitsfalle kann dann während des Spiels autonom und ohne explizite Kommunikation stattfinden, was zum einen in der Regel schneller ist und zum anderen nicht vom Gegner mitgehört werden kann.

living systems AG - Tel.: +49 (0)771/8987-0 - Fax: +49 (0)771/8987-100 – www.living-systems.com

Anhand der Fußballumgebung kann auch Kooperation auf Basis impliziter, d.h. durch Veränderung der Umgebung stattfindender Kommunikation, demonstriert werden. Beim Doppelpass wird beispielsweise durch das Passen des Balls und gleichzeitiges Losrennen des Agenten die Umgebung derart verändert, dass der angespielte Agent auch ohne explizite Kommunikation die ‚Nachricht' versteht und den Ball wieder dem ersten Agenten zurück passt.

Fazit

Agenten spielen in der Wissenschaft seit Mitte der achtziger Jahre eine Rolle. Mit der Definition von Fußball als Standardumgebung und der Einführung der RoboCup WM können verschiedene Ansätze und Agentensysteme direkt miteinander verglichen werden. Gleichzeitig können Agenten auch der nicht-Fachwelt näher gebracht werden. Abgesehen von der hervorragenden Eignung dieser Domäne, Autonomie und Kollaboration von Agenten zu demonstrieren, macht die Erstellung von Fußball-Agenten auch richtig viel Spaß!

Referenzen

[1] Informationen zum RoboCup stehen unter http://www.robocup.org
[2] Das Serverprogramm sowie ein Manual der Simulationsliga stehen unter http://sourceforge.net/projects/sserver/ zum download bereit.
[3] Stone, P., Balch, T., Kraetzschmar, G. (Eds.) (2001). *RoboCup 2000: Robot Soccer World Cup IV*. Springer-Verlag, Heidelberg.
[4] Veloso, M., Pagello, E., Kitano, H. (Eds.) (2000). *RoboCup-99: Robot Soccer World Cup III*. Springer-Verlag, Heidelberg.
[5] Asada, M., Kitano, H., (Eds.) (1999). *RoboCup-98: Robot Soccer World Cup II*. Springer-Verlag, Heidelberg.
[6] Kitano, H. (Ed.) (1998). *RoboCup-97: Robot Soccer World Cup I.* Springer-Verlag, Heidelberg.
[7] Dorer, K. (1999). *Behavior Networks for Continuous Domains using Situation-Dependent Motivations*. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, Seiten 1233-1238, Morgan Kaufmann, Cambridge, Mass.
[8] Rao, A. S. und Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. In R. Fikes und E. Sandewall (Eds.), *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning.* Seiten 473-484. Morgan Kaufmann, Cambridge, Mass.

living systems AG - Tel.: +49 (0)771/8987-0 - Fax: +49 (0)771/8987-100 – www.living-systems.com