Università degli Studi di Padova

Facoltà di Scienze Matematiche Fisiche e Naturali

Corso di Laurea Triennale in Informatica

Progetto per il corso Intelligenza Artificiale

Pianificazione Graphplan e simulazione RoboCup Soccer

Docente: Prof. Alessandro Sperduti

Studente: Tessaro Mikhalevich Alexej, 542123

Anno Accademico 2009 - 2010

Table of Contents

1. Introduzione

RoboCup è una iniziativa internazionale volta ad incentivare ricerca e progresso nei campi della **robotica** e dell'**intelligenza artificiale**.

L'obiettivo è la realizzazione di calciatori virtuali (*agenti*) capaci di competere all'interno di un ambiente *complesso*, *dinamico*, *incerto* e, nel caso di una vera e propria partita, *multi-agente*. Numerosi gruppi di ricerca hanno progettato le loro squadre utilizzando svariati approcci, arrivando ad ottenere giocatori dal comportamento e dalla strategia simili a quello dei calciatori umani.

L'ambiente RoboCup presenta le seguenti caratteristiche:

- ✓ **l'ambiente** è altamente **dinamico**
- ✓ la **percezione** di ogni giocatore è **localmente limitata**
- ✓ il ruolo di un giocatore può essere differente
- ✓ la **comunicazione** tra i vari giocatori è **limitata**, quindi ciascun agente deve sapere **adattarsi autonomamente** ed essere flessibile ai cambiamenti

Lo scopo del progetto è di architettare un sistema "*decisionale*" che permetta al giocatore virtuale di interpretare, decidere, ed agire sulla base delle informazioni locali a disposizione.

2. Struttura della Soluzione

L'ambiente di simulazione RoboCup è costituito dai seguenti componenti:

- ✓ *rcssserver* : **Soccer Server**, è il controllore della simulazione, impone le regole di gioco (file di configurazione) e i protocolli di comunicazione client-server.
- ✓ *rcssmonitor* : **Soccer Monitor**, rappresentazione visuale di gioco che monitorizza costantemente campo e giocatori.

I **client** sono *agenti autonomi indipendenti* che danno vita alle squadre di giocatori RoboCup.

Al fine di poter indirizzare il progetto verso lo studio del comportamento dei giocatori, è stato utilizzato **Atan**, un sistema di interfacciamento Java che astrae lo sviluppo del client dalle responsabilità di basso livello:

- ✓ *connessione* al Soccer Server tramite protocollo UDP
- ✓ *parsing* dei messaggi provenienti dal Soccer Server
- ✓ *generazione* dei messaggi riconoscibili dal Soccer Server

Soccer Server e interfacce Atan sono state modificate per poter modellare due condizioni di gioco semplici:

1. **Prima dinamica**

- ✓ Unico giocatore intelligente generato casualmente nella propria metà-campo.
- ✓ Metà-campo avversaria vuota.
- ✓ Palla al centro.
- ✓ *Obiettivo*: trovare il pallone, dirigersi verso la porta avversaria e fare goal.

2. **Seconda dinamica**

- ✓ Unico giocatore intelligente generato casualmente nella propria metà-campo.
- ✓ Metà-campo avversaria con n avversari statici (ostacoli).
- ✓ Palla al centro.
- ✓ *Obiettivo*: trovare il pallone, dirigersi verso la porta avversaria destreggiandosi attraverso gli ostacoli e fare goal.

L'intelligenza del singolo giocatore è stata modellata attraverso un semplice linguaggio di pianificazione **STRIPS-like**.

Costituito dai seguenti predicati:

- ✓ *can-see-ball* : il giocatore vede la palla
- ✓ *can-see-area* : il giocatore vede l'area di rigore
- ✓ *can-see-goal* : il giocatore vede la porta
- ✓ *have-ball* : il giocatore possiede la palla (può calciare)
- ✓ *clear-field* : il giocatore ha campo libero di fronte a sè
- ✓ *can-score* : il giocatore può calciare in porta
- ✓ *ball-in-net* : la palla è in goal

Che vanno a formare le seguenti azioni:

- ✓ *find-ball*
  precondizioni : []
  effetti positivi : [*can-see-ball*]
  effetti negativi : []

- ✓ *find-area*
  precondizioni : [*have-ball, clear-field*]
  effetti positivi : [*can-see-area*]
  effetti negativi : []

- ✓ *find-goal*
  precondizioni : [*have-ball, clear-field, can-see-area*]
  effetti positivi : [*can-see-goal*]
  effetti negativi : []

- ✓ *run-to-ball*
  precondizioni : [*can-see-ball*]
  effetti positivi : [*have-ball*]
  effetti negativi : []

- ✓ *run-to-goal*
  precondizioni : [*have-ball, clear-field, can-see-goal*]
  effetti positivi : [*can-score*]
  effetti negativi : []

- ✓ *dribble*
  precondizioni : [*have-ball*]
  effetti positivi : [*clear-field*]
  effetti negativi : []

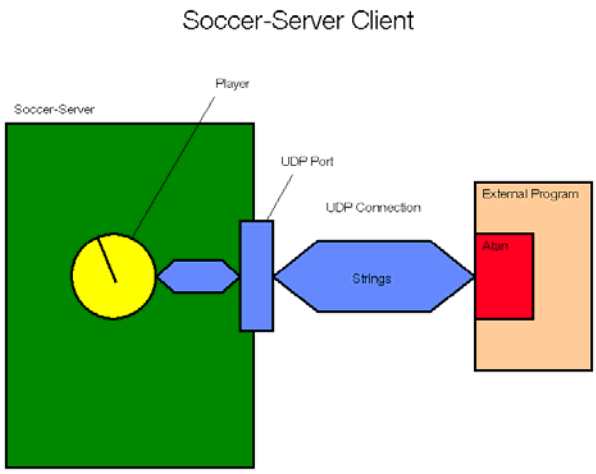
- ✓ *score-goal*
  precondizioni : [*have-ball, can-score*]
  effetti positivi : [*ball-in-net*]
  effetti negativi : []

Il Soccer Server genera una serie di percezioni locali (es. vede la palla, vede la porta, vede un avversario, vede l'area di rigore, ...) ad intervalli regolari (**ciclo di monitoraggio**). Queste percezioni (**sensori**), sono state utilizzate per formulare **obiettivi locali ordinati per priorità** ed inizializzare ed eseguire **PL-PLAN**, una serie di librerie Java che implementano diversi algoritmi di pianificazione, tra i quali **Graphplan**.

Il giocatore agisce quindi sulla base del piano risultante, conseguendo l'obiettivo corrente a massima prorità, e richiede la formulazione di un nuovo piano quando le percezioni sensoriali indicano una variazione di rilievo nel contesto delle informazioni percepite.

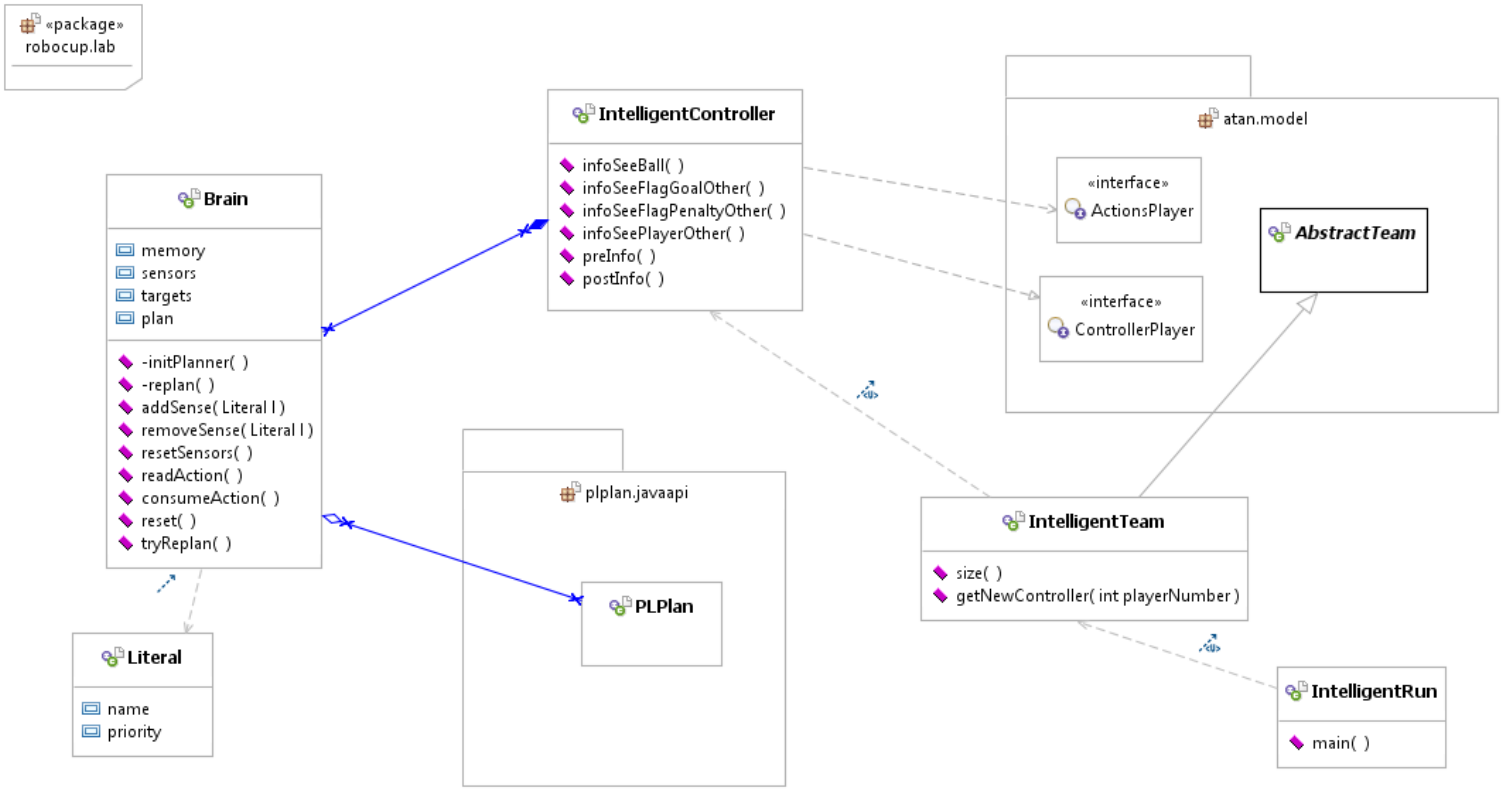
3. Rappresentazione della Soluzione

Atan permette di scrivere un Soccer Client Java interfacciandosi completamente al Soccer Server:



Per utilizzare Atan, la propria **classe Controller** (controllore del giocatore) deve implementare l'interfaccia *atan.model.ControllerPlayer*, che fornisce tutti i metodi necessari a monitorare lo stato corrente del giocatore.

Inoltre, una **classe Team** che estende la classe astratta *atan.model.AbstractTeam*, deve specificare il numero di giocatori della propria squadra, e il relativo Controller corrispondente a ciascun player.



IntelligentController

Responsabile del monitoraggio dello stato del giocatore sul campo. Implementa l'interfaccia *ControllerPlayer* del package *atan.model*.

Il ciclo di monitoraggio viene periodicamente eseguito ogni 100ms scatenando la seguente procedura:

Esecuzione del metodo *preInfo():* inizializzazione dei sensori.

Esecuzione dei metodi relativi alle percezioni del giocatore: registrazione delle percezioni.

Esecuzione del metodo *postInfo():* eventuale ripianificazione, e traduzione delle azioni di planning (**macro-azioni**) in azioni di gioco (**micro-azioni**).

IntelligentTeam

Responsabile della costruzione della squadra. Estende la classe astratta *AbstractTeam* del package *atan.model*. Associa un controller specifico ad ogni giocatore della squadra.

IntelligentRun

Responsabile dell'esecuzione del Soccer Client. Inizializza la squadra e connette ciascun giocatore al Soccer Server.

Brain

La mente intelligente di supporto al controller del giocatore. Responsabile dell'esecuzione opportuna del pianificatore e della memorizzazione dello stato corrente e passato dei sensori, detiene inoltre una *PriorityQueue* utile alla selezione dell'obiettivo locale di utilità massima.

Utilizza il pianificatore **PL-PLAN**, algoritmo di pianificazione **Graphplan**.

Literal

Rappresentazione di un predicato letterale semplice costituito da nome stringa e priorità intera.

4. Configurazione ed Esecuzione

RoboCup Soccer esegue sotto Windows e necessita di JDK 6.

Sono stati utilizzati i seguenti componenti e relative versioni:

- ✓ **rcssserver-14.0.2-win** (*http://sourceforge.net/projects/sserver/files/rcssserver/14.0.2/rcssserver-14.0.2-win.zip/download*)
- ✓ **rcssmonitor-14.1.0-win** (*http://sourceforge.net/projects/sserver/files/rcssmonitor/14.1.0/rcssmonitor-14.1.0-win.zip/download*)
- ✓ **atan_0.4.3** (*http://sourceforge.net/projects/atan1/files/Atan/v0.4.3/atan_0.4.3.zip/download*)
- ✓ **plplan-0.43** (*http://plplan.philippe-fournier-viger.com/plplan-0.43.jar*)

Per eseguire i due scenari di progetto, procedere con le seguenti istruzioni:

1. Lanciare il Soccer Server (/rcssserver-14.0.2-win/rcssserver.exe)
2. Lanciare il Soccer Monitor (/rcssmonitor-14.1.0-win/rcssmonitor.exe)
3. Lanciare solo (primo scenario) il JAR Intelligente (/IntelligentTeam.jar)
4. Lanciare anche (secondo scenario) il JAR Avversario (/OpponentTeam.jar)
5. Ad ogni goal è necessario forzare il playmode a play_on, dal Soccer Monitor: Referee ->Change Playmode->play_on

5. Conclusioni e Possibili Estensioni

Ci si accorge della complessità nel progettare squadre di simulazione RoboCup Soccer quando si decide di provarci. La percezione locale del giocatore e la moltitudine di fattori da prendere in considerazione nella scelta dell'azione da eseguire (nel rispetto dei **vincoli temporali**) determina l'utilizzo di algoritmi ed euristiche altamente sofisticati.

Utilizzare un pianificatore allo scopo di determinare la migliore azione in un certo istante risulta possibile anche se, al fine di produrre agevolmente un buon piano dovrebbero valere le seguenti assunzioni:

- ✓ Agente come sola causa di **cambiamento** dell'ambiente.
- ✓ Agente avente conoscenza completa dello stato dell'ambiente . (**omniscenza**)
- ✓ Tutto ciò che è vero, incluso nella descrizione dello stato corrente. (**Closed World Assumption**)
- ✓ **Azioni deterministiche**, ovvero nessuna incertezza nel loro effetto.

Tutte caratteristiche che l'ambiente RoboCup non è in grado di garantire, ma, se un agente non ha accesso ad informazioni esaustive riguardo allo stato dell'ambiente, dovrebbe optare per un'azione che possa quantomeno ridurre la sua incertezza.

L'articolo "Extending Graphplan to Handle Uncertainty & Sensing Actions" [1] descrive **Sensory Graphplan** (*SGP*), un discendente dell'algoritmo qui utilizzato che si propone di risolvere problemi di pianificazione legati alla contingenza.

[1] [Weld, Anderson & Smith, 1998] Extending Graphplan to Handle Uncertainty & Sensing Actions In. *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*