Especificación de Requerimientos de Software

26 de noviembre de 2006

Versión C1.0.1

Histórico de revisiones

Fecha	Versión	Descripción
20/02/2006	E1.0.0	Primera versión de la Especificación de Requerimientos Funcionales del sistema FIbRA
24/02/2006	E1.0.1	Se agregan sub-requerimientos para indicar distintas posibilidades de funcionamiento del siste
03/03/2006	E1.0.2	Revisión del documento y compatibilización con documento de alcance del sistema
31/03/2006	C1.0.1	Corrección de Referencias

Facultad de Ingeniería

Universidad de la República Oriental del Uruguay

Índice

1. Introducción

1.1. Propósito

Se presentan los requerimientos funcionales del sistema FIbRA, así como los usuarios que harán uso del mismo. Los requerimientos no funcionales pueden encontrarse en el documento Requerimientos Suplementarios [4]. Ambos documentos capturan la totalidad de requerimientos del sistema.

1.2. Alcance

Como lo indica el documento Modelo de Casos de Uso [5], el sistema ofrece una única funcionalidad que puede ser utilizada por los actores (agentes externos al sistema). Esta funcionalidad consiste en determinar las velocidades izquierda y derecha de las ruedas de cada robot del equipo, necesarias para implementar una estrategia. El proceso de toma de decisiones tiene como principal y único insumo el estado actual del juego que es recibido desde afuera del sistema. Sin embargo, no existe una única forma de procesar esta información. En lo sucesivo a estas formas de procesamiento o "razonamiento" que se puede realizar sobre la información, le llamaremos "habilidades" o "capacidades".

Este documento describe cada una de las habilidades que serán implementadas, al mismo tiempo que presenta, en los casos de habilidades investigadas pero que no serán implementadas, una breve discusión que justifica la decisión de no incluirlas en el alcance del sistema a construir.

1.3. Definiciones, Acrónimos y Abreviaciones

1. FIRA : Federation of International Robot-soccer Association
2. SimuRoSoT : Simulated Robot Soccer Tournament - Sumilador que utilizará el sistema a construir.
3. DLL : Dynamic Link Library
4. LINGO : Scripting language para implementación de estrategias para SimuroSot
5. RoboSoccer : Robot Soccer - Fútbol de Robots.

2. Descripción del sistema

2.1. Introducción

El sistema FIbRA consiste en un equipo de Robosoccer capaz de competir en la categoría SimuroSot de la FIRA [2].

Para lograr mejorar las habilidades del equipo, se buscan aplicar técnicas de Aprendizaje Automático que le permitan adaptar su estrategia al juego del oponente. Algunas de las posibilidades incluyen el uso de Redes Neuronales para la detección de secuencias de pases, el uso de Aprendizaje por Refuerzo para ajustar la toma de decisiones del equipo o el modelado con una arquitectura Multi-Agente. En este último caso, los agentes no necesariamente deben pensarse como metáforas de los robots (jugadores), sino que podrían representar otro tipo de entidades. Por ejemplo, cada agente podría captar distitintos aspectos del juego (ya sea el propio o el contrario) para contar con distintas visiones y generar meta-información. Esta meta-información podría ser utilizada por la estrategia para mejorar la toma de decisiones.

3. Usuarios del sistema

Dadas las propias características del trabajo, el sistema es utilizado por los desarrolladores y los tutores del proyecto. El sistema es construido como parte de la investigación en el área de fútbol de robots realizada por los desarrolladores.

4. Requerimientos funcionales

4.1. Devolver velocidades

Se reciben del simulador los datos que describen el estado actual del juego (*Environment*). Esta información contiene, entre otros datos, las posiciones de los robots (propios y oponentes), la posición de la pelota, quién posee la pelota actualmente, etc. En base a esta información, la estrategia del sistema calcula las mejores acciones a llevar a cabo para las condiciones dadas y define las velocidades que se deben asignar a cada rueda de cada robot propio para lograr dichas acciones. Se retorna al simulador el conjunto de velocidades que se asignan a las ruedas de cada robot.

Como se mencionó en la sección [1.2], existen varias formas de lograr este requerimiento. En las sub-secciones siguientes se destacan y describen las habilidades candidatas a formar parte del sistema. En el documento [3] se justifican las decisiones que determinaron la inclusión o exclusión de estas habilidades en el alcance.

4.1.1. Detección de patrones de juego del equipo contrario

Esta habilidad o capacidad refiere a intentar predecir el comportamiento del equipo contrario para poder anticipar sus jugadas, mejorar la defensa y eventualmente el ataque del equipo. Para obtener esta habilidad es necesario, primeramente, analizar el juego del equipo contrario durante un cierto periodo de tiempo.

Al comienzo del juego el sistema utilizará una estrategia de juego determinada mientras recolecta la información necesaria para poder iniciar la detección de patrones. Una vez recolectada la cantidad suficiente de información, la misma es procesada. Como resultado se obtiene información enriquecida (meta-información) que el sistema podrá utilizar, en lo sucesivo, para mejorar su estrategia.

Por lo tanto, una vez procesada toda la información recolectada, el equipo tiene la posibilidad de utilizar estos datos en la toma de decisiones.

En las secciones subsiguientes se describen dos estrategias de predicción que tienen como objetivo predecir dos aspectos que hacen al juego del equipo contrario y que pueden ser utilizadas en conjunto o por separado.

Detectar la formación del equipo contrario

Busca determinar el sistema de juego o la formación del equipo contrario. Esto implica determinar cuántos robots se utilizan para defensa cuando el equipo esta siendo atacado, cuántos robots juegan como atacantes cuando el equipo realiza jugadas de ataque, las zonas de la cancha donde se ubican los jugadores, etc.

Con esta información se pretende dotar a la estrategia de una herramienta que mejore su propia formación dentro de la cancha adecuándola al rival de turno y/o mejore los movimientos para realizar jugadas de defensa o ataque.

Detectar zonas de mayor actividad del equipo contrario

Se intenta reconocer las áreas de mayor influencia o actividad de los jugadores del equipo rival, así como la secuencia de pases más probable entre dichas zonas de influencia. En cierta forma se busca detectar patrones de cooperación o colaboración entre los jugadores del equipo contrario al momento de realizar jugadas de ataque y eventualmente defensa.

Para lograr esto, se analizan las zonas por donde se detecta mayor tránsito de robots contrarios y los pases que éstos realizan. Para lo primero, se sigue el rastro de todos los robots contrarios, sin identificarlos individualmente, y se almacena dicha información durante un tiempo determinado. Para lo segundo, en paralelo, se almacena la secuencia de pases detectada cuando el equipo contrario tiene el dominio de la pelota y se encuentra en zona de ataque.

Al finalizar la recolección de estos datos, se cuenta con un mapa de juego del contrario que permite determinar las zonas más utilizadas por el equipo contrario (zonas calientes) para atacar y los pases realizados en dicho ataque.

Detectar estrategia o jugadas del equipo contrario

En la sección [4.1.1] se describió la estrategia de predicción que se utiliza para detectar zonas de actividad y pases del equipo contrario. En esta sección se describe la forma en que se espera predecir secuencias de pases como parte de la estrategia del oponente, utilizando la información de zonas y pases.

Una vez detectadas las zonas de ataque del equipo contrario y los pases realizados en situaciones ofensivas, se procesa esta información, combinando el resultado de ambos análisis, generando meta-información que contenga la relación de precedencia entre las zonas identificadas. Esta relación indicaría el destino probable de un pase probable desde una región en la cancha donde el oponente tiene la pelota.

El resultado puede verse como un mapa de juego del contrario, donde se indican las zonas de influencia del contrario, las secuencias de pases realizados entre dichas zonas y la frecuencia (o probabilidad) con la que se realizan estos pases entre una zona y otra. También puede ser visto como un grafo que conecta las zonas de influencia mediante pases realizados en alguna jugada de ataque del equipo contrario. Cada arista de este grafo (pase realizado) contiene un peso que indica la probabilidad de ocurrencia de dicho pase (en base a la frecuencia de pases realizados en la etapa de recolección de información).

4.1.2. Evolución de jugadas o equipos

Obtener un equipo cuya estrategia de juego o comportamientos de grupo sean evolucionados utilizando técnicas de aprendizaje automático basadas en la evolución. Una posible técnica a utilizar es Programación Genética, mediante la cual se puede obtener un equipo cuya estrategia o comportamiento sean evolucionados partiendo de las habilidades básicas de cada jugador.

4.1.3. Sistema Multi-Agente

Implementar el sistema como un conjunto de agentes que cooperan entre sí para resolver el problema de ganarle a un equipo rival.

Para resolver dicha implementación se puede aplicar el pradaigma de Sistemas Multi-Agentes de varias formas:

1. utilizar un agente por cada robot, otorgándole a cada robot la capacidad de comunicarse con los demás robots de su equipo, y de esta forma cooperar entre ellos para lograr un objetivo común.

2. desarrollar un sistema multi-agentes donde cada agente se relaciona con distintos aspectos del juego, y no con cada jugador. Cada agente es visto como un observador del juego y cada uno de ellos es capaz de obtener y procesar información útil para el equipo en su conjunto. Una vez que cada agente obtuvo la información que le compete, todos los agentes comparten dicha información para lograr un consenso de grupo y tomar una decisión sobre las acciones que debe realizar cada robot para lograr el objetivo del equipo. En este caso, se debe definir un lenguaje de comunicación entre agentes mediante el cual compartirán la información.

3. Complementando el punto anterior, se puede establecer un sistema de creencias, donde cada agente tiene un factor de credibilidad, el cual es actualizado a lo largo del juego en base a

recompensas por los buenos resultados obtenidos a partir de sus acciones propuestas. En este caso podría utilizarse alguna técnica de Aprendizaje por Refuerzo para ir ajustando los factores de credibilidad de cada agente.

4.1.4. Reorganización del equipo

Dotar al equipo de la habilidad de reorganizarse dinámicamente es una idea que no es nueva y que ha tenido cierto éxito. La idea detrás de esta habilidad es que el sistema sería capaz de variar su sistema de juego en base a resultados intermedios y datos del juego. El equipo comienza el juego utilizando un sistema de juego con una formación inicial determinada, donde cada robot tiene una zona de juego delimitada. Con el transcurso del tiempo, el sistema podría ajustar su sistema de juego o las zonas de influencia de cada robot para contrarrestar las características de juego del equipo contrario o adaptarse a los resultados. Este ajuste del equipo puede ser realizado utilizando algoritmos de aprendizaje por refuerzo como *Q-Learning*.

5. Interfaz de usuario

El sistema no interactúa directamente con el usuario final, sino que lo hace indirectamente a través del simulador recibiendo posiciones del mismo y enviando la información de las velocidades como respuesta. Es dicho simulador el encargado de la interacción con el usuario final. Las modificaciones a las posiciones que el sistema recibe para tomar decisiones sólo pueden ser modificadas a través del simulador.

6. Interfaz con software

El simulador de la FIRA en su versión 1.5 establece una restricción importante con respecto a la interfaz que debe ofrecer la estrategia a implementar. El sistema debe cumplir dicha interfaz para poder interoperar con el simulador [1, 2].

El SimuroSot ofrece dos posibilidades para la interfaz de comunicación:

1. Indicar un archivo de texto escrito en lenguaje LINGO, conteniendo toda la estrategia del sistema.
2. Indicar al simulador que la estrategia va a ser invocada a través de una DLL escrita en C++.

Para el segundo caso, la librería DLL debe cumplir con una interfaz definida por el simulador y detallada en la ayuda del mismo o en el documento de Especificaciones para SimuroSot [1].

La DLL escrita en C++ debe implementar tres métodos que serán invocados por el simulador:
- extern "C" STRATEGY_API void Create (Environment *env);
- extern "C" STRATEGY_API void Strategy (Environment *env);
- extern "C" STRATEGY_API void Destroy (Environment *env);

El método Create es invocado una única vez al iniciar un nuevo juego, indicándole al sistema que comienza un nuevo partido. Este método recibe el primer Environment generado por el simulador, con las posiciones iniciales de los robots y la pelota, y ofrece la posibilidad de que el sistema inicialice sus variables y secciones de memoria para llevar adelante un partido.

El método Strategy será invocado por el simulador aproximadamente 60 veces por segundo, pasando por referencia el Environment con las posiciones actuales de juego y esperando en el mismo las velocidades resultantes del sistema.

El método Destroy es invocado una única vez por partido, cuando se finaliza un juego, pasando como parámetro el último Environment del partido. Esta función permite al sistema detectar un fin de juego y liberar la memoria que sea necesaria liberar.

Referencias

[1] Gustavo Armagno, Facundo Benavides, and Claudia Rostagnol. Especificaciones para simurosot. Technical report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2005. 8

[2] Gustavo Armagno, Facundo Benavides, and Claudia Rostagnol. Reglas para simurosot. Technical report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2005. 5, 8

[3] Gustavo Armagno, Facundo Benavides, and Claudia Rostagnol. Alcance del sistema. Technical report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2006. 6

[4] Gustavo Armagno, Facundo Benavides, and Claudia Rostagnol. Especificación suplementaria de requerimientos. Technical report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2006. 5

[5] Gustavo Armagno, Facundo Benavides, and Claudia Rostagnol. Modelo de casos de uso. Technical report, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay, 2006. 5