Planificador continuo como controlador de agentes robots

Mario Moya Claudio Vaucheret
Departamento de Ciencias de La Computación
Facultad de Economía
Universidad Nacional del Comahue
email: {moya.mario,vaucheret}@gmail.com

14 de marzo de 2008

Resumen

En este trabajo presentamos un planificador continuo que implementa un controlador para un equipo de futbol de robots. El planificador continuo generaliza el planificador de orden parcial implementado en Prolog.

1. Introducción

El concepto de controlador en teoría de controles es idéntico al de un agente en Inteligencia Artificial. El trabajo de la Inteligencia Artificial es diseñar el programa agente, que implemente la función de mapear percepciones a acciones: tomara la percepción actual como entrada desde los sensores y devolverá una acción a los efectores o actuadores[1].

En general podemos encontrar algunos tipos básicos de programas de agentes que engloban los principios básicos de los todos los sistemas inteligentes.

Tenemos por un lado los agentes puramente reactivos, los mas simples, que seleccionan acciones sobre las bases de las percepciones actuales ignorando percepciones del pasado. Estos agentes reactivos funcionan con reglas de *condición-acción*, y cuando ninguna condición se cumple tienen una acción por defecto. Tienen la ventaja de ser muy simples de programar pero la gran desventaja de de tener una muy limitada pro-actividad.

Un agente se denomina deliberativo cuando representa su estado interno como una base de conocimiento y utiliza un sistema deductivo para generar las acciones. Conocer el estado del ambiente no siempre es suficiente para decidir que se debe hacer. Los *agentes orientados a metas*, necesitan alguna información que describa las situaciones deseables. Estos agentes implementan búsqueda o planificación para encontrar secuencias de acciones que lo lleven a alcanzar la meta deseada.

En algunas situaciones es necesario un agente híbrido, dándole la capacidad de que su comportamiento sea reactivo y pro-activo, descomponiendo su arquitectura en capas, donde cada capa es un subsistema que trata con cada uno de estos comportamientos, y logra la interacción entre ellos.

En este trabajo optaremos por otra aproximación. Presentaremos la implementación de un controlador que a la vez es un planificador continuo. Este controlador, además de su carac-

[1] en inglés, *actuators*.

terística reactiva tiene la capacidad de mantener planes parciales y modificarlos de acuerdo a la información variante del entorno logrando un comportamiento pro-activo.

El ambiente donde queremos aprovechar las características de este controlador es en el problema del fútbol con robots [6].

En las siguiente sección presentamos la planificación continua implementada como una generalización de los planificadores de orden parcial Luego mostraremos un ejemplo de este controlador en el ambiente de fútbol con robots. Por último presentamos las conclusiones, el trabajo en progreso y el trabajo futuro.

2. Planificación Continua

Planificación esta considerada como una de las áreas mas importantes en la inteligencia artificial. El objetivo de un planificador encontrar una secuencia de acciones que lleven a una solución válida a un problema. Es necesario contar con un lenguaje de representación que sea lo suficientemente expresivo para poder describir el dominio del problema, las acciones y como estas afectan el ambiente. El más popular de ellos es STRIPS, pero hay variantes mas expresivas como Planning Domain Description Language o PDDL[2] que intenta ser un estándar en la representación de este tipo de problemas.

Uno de los planificadores más populares es el de orden parcial[4], que saca ventaja de la descomposición de un problema en submetas. Tiene un carácter incremental, donde el algoritmo va completando un plan parcial resolviendo sus deficiencias.

El algoritmo[8] comienza con un plan vacío, conteniendo sólo las acciones Start y Finish. Start es una acción sin precondiciones, para que pueda ser ejecutada inmediatamente, y tiene como efectos las condiciones que describen el estado inicial del problema. De manera análoga Finish sin efectos y tiene como precondiciones los literales que describen la meta a alcanzar.

Además cada plan tiene un conjunto de restricciones de ordenamiento que establecen que acción debe ejecutarse en antes que otra, sin necesidad de que este inmediatamente antes.

Uno de los principales componentes del algoritmo es el tratamiento de los enlaces causales, del tipo $A \xrightarrow{p} B$, que registra que p es un efecto de la acción A necesaria para la acción B. Las precondiciones abiertas son las que aún no son resueltas por ninguna acción. El algoritmo de planificación intentará resolverlas insertando nuevas acciones que las satisfagan sin introducir conflictos[7].

El ambiente del fútbol de robots es sumamente dinámico provocando que un planificador clásico no resulte satisfactorio debido que los planes correctos se vuelven inconsistentes e inaplicables en el momento de su ejecución. Una alternativa es que planificador sea continuo[5], donde los planes sean artefactos sin terminar que deben evolucionar en un entorno siempre cambiante.

El Planificador de Orden Parcial es ideal para ser generalizado como un planificador continuo pues puede ser visto como un algoritmo removedor de fallas, donde las fallas son precondiciones abiertas y conflictos causales. El planificador continuo además abarca otro tipo de fallas: el agente puede decidir agregar nuevas metas al estado Finish, puede agregar un enlace causal a una precondición abierta, eligiendo una nueva acción que lo resuelva, puede poner restricciones en el orden de ejecución de las acciones para evitar los conflictos, puede remover enlaces ya no soportados en Start previniendo la ejecución de acciones cuyas precondiciones ya son verdaderas. Además puede remover acciones redundantes en el plan y generar nuevas metas o precondiciones abiertas. También puede ejecutar acciones sin precondiciones abiertas que no tengan otras antes en el ordenamiento.

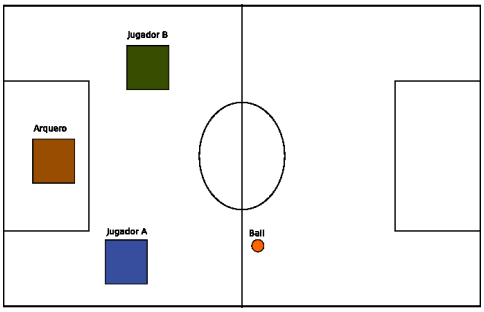

Figura 1: Estado Inicial

De esta manera el planificador puede funcionar como el controlador de un agente. En la sección siguiente ejemplificaremos su funcionamiento.

3. Un Ejemplo de Fútbol con Robots

Ejemplificaremos el uso del planificador continuo como controlador de los agentes de un equipo de fútbol con robots.

Algunas de las primitivas dentro de la base de conocimiento que necesitamos para describir el dominio son:

- $At(R, X)$ Verdadero si el jugador R está en la posición X.
- $HasBall(R)$ Verdadero si el jugador R tiene la pelota.
- $IsAt(X, Y)$ Verdadero si el objeto X está en la posición Y.
- $OpponentArea(R)$ Verdadero si R está en el área del equipo contrario.

También tenemos algunos predicados derivados como $InReach(R, X)$, que es Verdadero si el objeto X esta dentro de la distancia que puede alcanzar pateando el jugador R.

Las acciones que necesitaremos son descriptas por los siguientes esquemas:

La acción de mover al robot sin la pelota hasta una posición. Acción(GoTo(Robot,TargetPosition), PRECOND: (¬ HasBall(Robot)), EFFECT: At(Robot,TargetPosition)).	La acción de mover al robot sin la pelota para hasta la pelota. Acción(GoToBall(Robot), PRECOND: (¬ HasBall(Robot)), EFFECT: HasBall(Robot) ∧ IsAt(Ball,Robot)).
La acción de pasar la pelota de un robot a otro. Acción(Pass(Robot1,Robot2), PRECOND: (HasBall(Robot1) ∧InReach(Robot1,Robot2)), EFFECT: (¬ HasBall(Robot1) ∧ IsAt(Ball,Robot2))).	La acción de que Robot1 espera un pase de Robot2, y obtiene la pelota. Acción(CatchPass(Robot1,Robot2), PRECOND: (IsAt(Ball,Robot1) ∧ InReach(Robot2,Robot1)), EFFECT: (HasBall(Robot1))).

El estado inicial de la cancha y los jugadores se muestra en la figura 1

Supongamos que un agente tiene la meta de obtener el estado $OpponentGoalArea(B) \wedge HasBall(B)$ y comienza a planificar para alcanzarla.

El plan se construye de manera incremental, se actualiza en cada ciclo de percepción y acción. Si ninguna acción es la mínima con sus precondiciones satisfechas el agente devuelve NoOp como su acción y percibe nuevamente. Asumimos en el ejemplo que el agente construye el plan de la figura 2.

El plan está completo, de acuerdo a él el controlador debería ejecutar las acciones $GotoBall(A)$ y $Goto(B, OpponentArea)$, pero antes de ejecutar estas acciones, la suerte interviene, y un factor externo pone la pelota en movimiento y llega hasta la posición del jugador A.

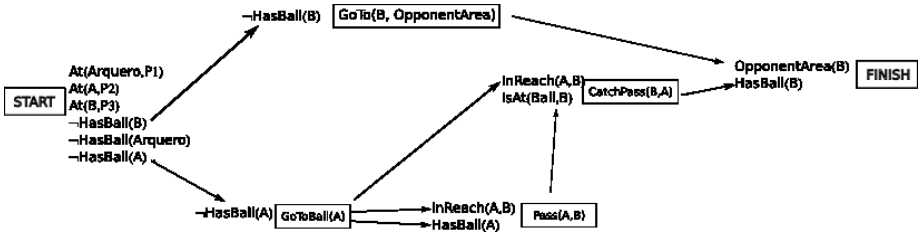

Figura 2: plan inicial

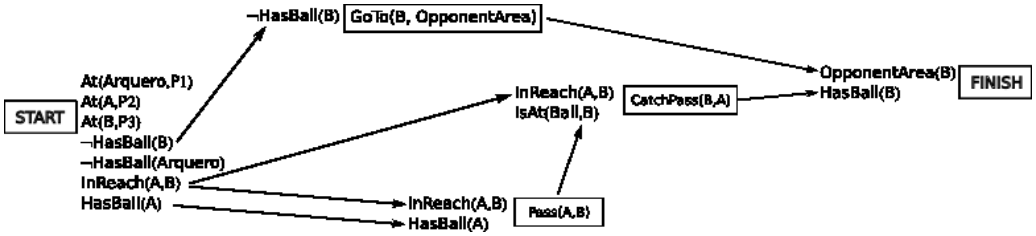

Figura 3: plan después de que una contingencia hace que A reciba la pelota

El agente percibe que la posición de la pelota ha cambiado y se da cuenta que el estado inicial es distinto. Actualizando su modelo al estado actual. Ahora el enlace causal [2] provisto por la precondición $\neg HasBall(A)$ ya no es necesaria y es removido del plan junto con la acción $GotoBall(A)$.

El agente saca ventaja de esta situación y los enlaces causales $GoToBall(A) \overset{IsAt(Ball,B)}{\longrightarrow} CatchPass(B,A)$ y $GoToBall(A) \overset{HasBall(A)}{\longrightarrow} Pass(A,B)$ pueden ser reemplazados por un enlace directo desde $START$ a $CatchPass(B,A)$ y $Pass(A,B)$ respectivamente. Extender estos enlaces causales no provoca ningún conflicto.

El nuevo plan se muestra en la figura 3. Supongamos ahora que $GoTo(B, OpponentArea)$ ya se ejecuto y $Pass(A,B)$ está lista para disparar, entonces puede ser removida del plan y ejecutarse. Desafortunadamente, el resultado no es el esperado y el jugador falla al intentar el pase apenas golpeando la pelota haciendo que quede alejada de los dos jugadores.

El nuevo estado mostrado en la figura 4 presenta precondiciones abiertas que tienen que ser resueltas y, por lo tanto, no puede ejecutar ninguna acción. Nuevamente la acción $Pass(A,B)$ satisface la precondición $IsAt(Ball,B)$ y esta acción a su vez tiene como precondición abierta $HasBall(A)$ y se repite el proceso para satisfacerla con la acción $GoToBall(A)$. Lo que lleva al plan de la figura 5

Esta vez la ejecución de todas estas acciones tienen éxito y se alcanza la meta deseada. A medida que se ejecutan las acciones y son eliminadas del plan se llega a un estado donde la meta *OpponentArea(B)* $\wedge$ *HasBall(B)* se satisface directamente desde START a FINISH. Entonces el agente está listo para formular nuevas metas.

[2]En ingles link causal

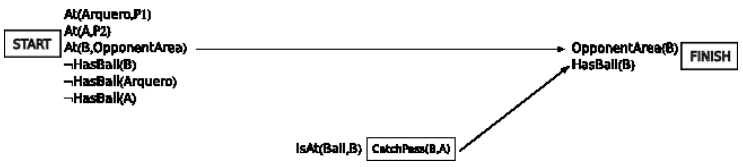

Figura 4: plan después de fallar el pase

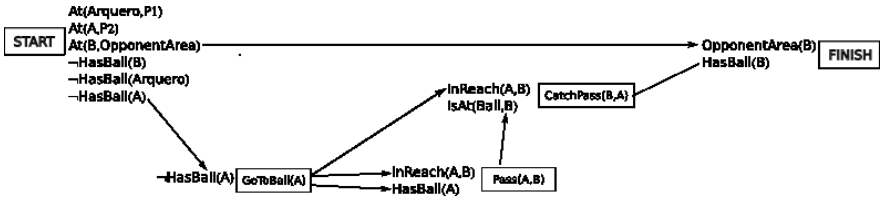

Figura 5: Antes de intentar otra vez el pase

4. Conclusión

Un controlador con las características mostradas en este trabajo se encuentra en estado de implementación. Hemos implementado en Prolog el planificador de orden parcial y se lo está extendiendo a su versión continua. Para ello se construyó un interprete del lenguaje de representación PDDL, y se adaptará el planificador para que entienda esta representación, y así poder probar su rendimiento en el ambiente del fútbol con robots.

Hemos estado trabajando en la implementación de Rakiduam, un equipo de fútbol con robots, que participó en las competencias de CAFR de las ediciones 2006 y 2007 en categoría simulados[3], y pretende participar en la edición 2008 que se realizará en agosto con cede en la Universidad Nacional del Comahue.

Rakiduam esta implementado en ciao prolog[1]. Es software libre, está liberado con una licencia GNU GPL y se puede obtener de `http://code.google.com/p/rakiduam`.

Referencias

[1] F. Bueno, D. Cabeza, M. Carro, M. Hermenegildo, P. López, and G. Puebla. *The Ciao Prolog System - A Next Generation Multi-Paradigm Programming Environment.* Grupo Clip, http://www.ciaohome.org/, the ciao system documentation series edition, August 2004.

[2] M. Ghallab, A. Howe, C. A. Knoblock, and McDermott D. Pddl-the planning domain definition language. Technical report, Yale Center for Computational Vision and Control, New Haven, Connecticut, 1998.

[3] P. Kogan, M Moya, G. Torres, and et al. Rakiduam: Un equipo de futbol con licencia gnu general public license. In *CAFR 2007, Campeonato Argentino de Futbol de Robots*, Buenos Aires, Agosto 2007. Caeti-Centros de Altos Estudios en Tecnología Informática.

[4] D. A. McAllester and D. Rosenblitt. Systematic nonlinear planning. In AAAI Press, editor, *Ninth National Conference on Artificial Intelligence (AAAI-91)*, volume 2, pages 634–639, Anaheim, California, 1991.

[5] Karen Myers. Toward a framework for continuos plannig and execution. In *AAAI Fall Symposium on Distributed Continual Planning*, 1998.

[6] Federation of International Robot-Soccer Association. Fira official website. `http://www.fira.net/`, 2008.

[7] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall Series in Artificial Inteligence. Prentice Hall, second edition, 2003.

[8] S. Soderland and D. S. Weld. Evaluating non-linear planning. Technical report, University of Washington Department of Computer Science and Engineering, Seattle, Washington, 1991.