LTU Blue Devils

4-Legged League

Technical Report

Team: Emily Trudell, Anthony Mitchell, Joseph Szostek
Advisor: Dr. Chung
chung@ltu.edu

Mathematics and Computer Science Department,
Lawrence Technological University
21000 West Ten Mile Rd.
Southfield, MI 48075
United States of America

http://ltu164.ltu.edu/aibo

4 May 2007

Abstract. This paper presents the work of the LTU Blue Devils for RoboCup 2007 for the 4-Legged League. The LTU Blue Devils are using the Tekkotsu [2] framework from Carnegie Melon University as a general base for AIBO [1] programming. The current team has continued working with last year's soccer behavior and has made great improvements. Before this year only basic attacker and goalie behaviors were completed. Now localization, wireless communication, multi-agent collaboration and fuzzy logic have been implemented into to this year's soccer behavior. Also the teams work on the attacker and defender's state machines will be presented along with the development of custom soccer motions. Also the development tool, created by last year's team, PyTekkotsu will be discussed.

Contents

1. Introduction

The LTU Blue Devils have participated in the Robocup 2005 US Open in Atlanta, Georgia. We also participated in the RoboGames in San Francisco in 2006. Our team is currently focused on the areas of localization, wireless communication, multi-agent collaboration and fuzzy logic. We have also worked on development tools like PyTekkotsu and written tutorials aimed at high school level students. We are using the Tekkotsu framework from Carnegie Melon University as a general base for AIBO programming. The tasks in the RoboCup competition, as well as technical challenges, require the Tekkotsu platform to be extended and enhanced in several critical areas. Although Tekkotsu is a platform to aid AIBO development it is designed for general programming, not explicitly for soccer. As such, we do not use any code directly from any other team or directly from Tekkotsu. All of our soccer behaviors are our original code, utilizing the Tekkotsu layer on top of Open-R.

2. State machine

We currently have the appropriate game state machine set up according to the RoboCup rules. Each type of player has its own state machine within the playing state. The attackers have five states as can be seen in Figure 1. The attacker's search state rotates the AIBO until the ball is found. The AIBO chases the ball in the track state. The wait state can only be reached if there are two attackers connected wirelessly. One attacker will wait for the other attacker to complete its action with the ball to keep them from colliding. The spin state was created to allow the dog to move with the ball to a good kicking position. It then performs a straight forward kick after finding a space undefended by the goalie.

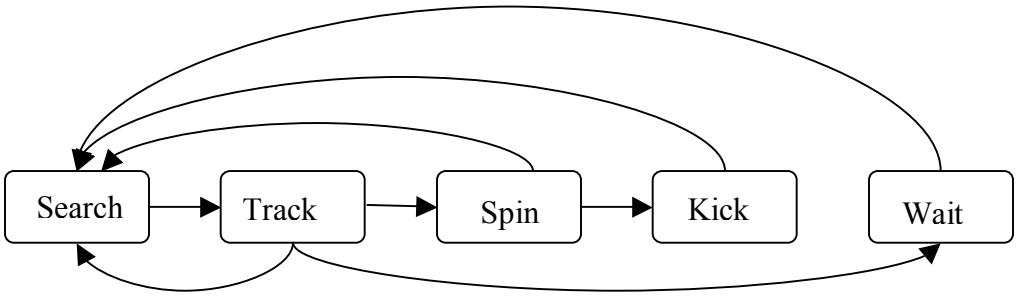

Fig. 1 The attacker's state machine.

The defender has four states. The lost state can only be reached if the dog is not close enough to the ball to need to defend its goal. The defender's search state differs from the attacker's search state because it does not constantly rotate and it does not immediately move into the track state once the ball is found. It waits until the ball is close enough to the home goal to begin pursuit. In the lost state the dog uses localization to reset itself in an appropriate defensive position on the field. The state machine for the defender can be seen in Figure 2.

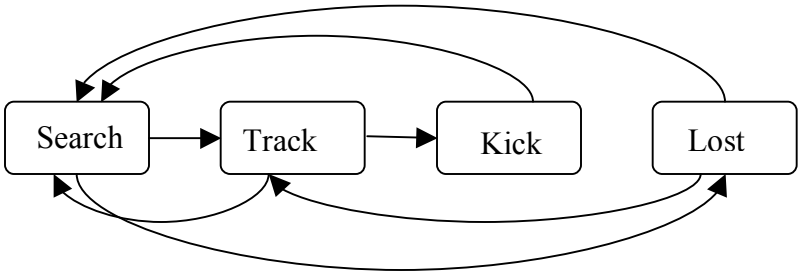

Fig. 2 The defender's state machine.

This state machine was first created by last year's team and has been refined and added to by this year's team.

3. Motion

3.1. Created Motions

Many custom motions have been created to perform soccer related tasks. A head butt motion to kick the ball about half the field length has been created. The development of this motion was done by team members using Sony's Medit tool. The motion has been refined many times over and currently seems to be our most efficient kick. We also have two kicks that allow the dog to kick the ball a short distance 90 degrees right or left. Lastly, a spin motion has been created that allows the dog to hold the ball for a short time and try to center itself on the goal. We found the development of this motion to be the hardest. Because we were only using Medit to create the motion, it took many tries on our part to get the dog to grip the floor while still moving left or right. Currently there is still room for improvement in this motion but it does allow the dog to grab and move the ball while still allowing the dog to get a clear view with its camera.

3.2. Fuzzy Ball Tracking

A fuzzy logic engine has been implemented into our ball tracking method. This smoothes out our ball tracking and makes it more reliable. The current fuzzy sets and rules are defined in Figures 3 through 5. The fuzzy engine consists of three C++ classes: Fuzzy set, rules, and fuzzy engine. These classes allow us to define simple two variable rules or more complex three or four variable rules. The singleton method was used to defuzzify the fuzzified inputs. This proved to be accurate enough to improve the ball tracking greatly. Using fuzzy logic has allowed us to use the ball area to directly affect the AIBO's speed without having to worry about oscillation caused by camera noise. It also allows us to directly relate the ball area to the output speed when distance from the ball and ball area do not have a simple linear relationship. Based on the ball area value a degree of membership is calculated for the fuzzy sets using the rules seen in Figure 5.

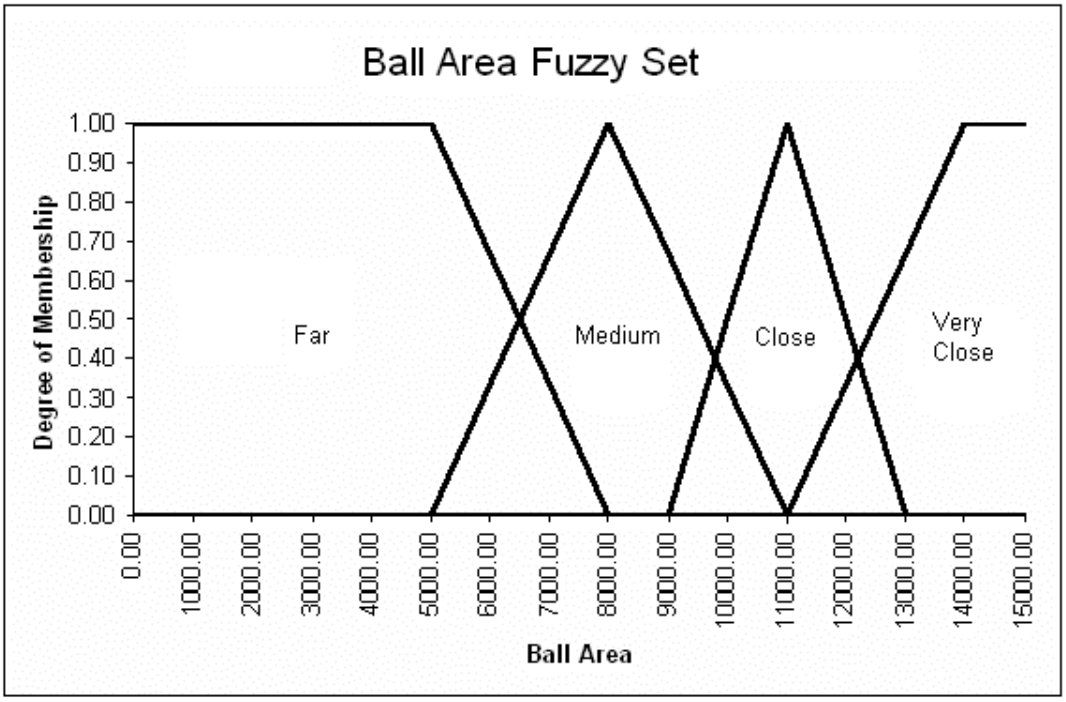

Fig. 3 The *ball area fuzzy set* used in the attacker's ball tracking.

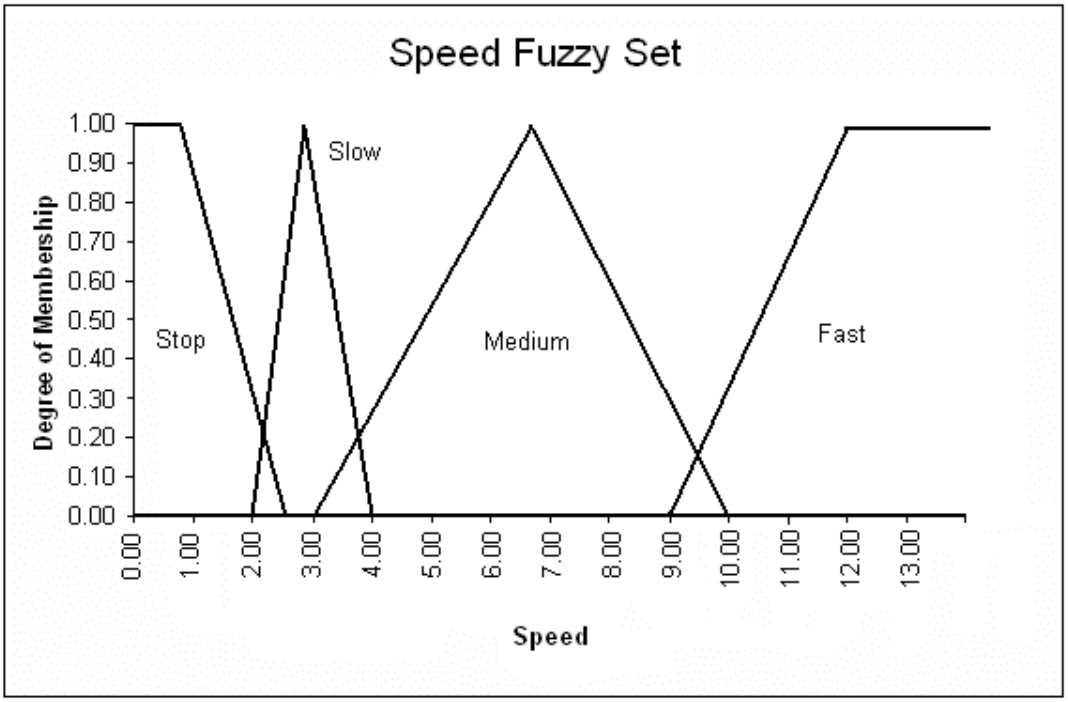

Fig. 4 The *speed fuzzy set* used in the attacker's ball tracking.

Rule 1:
IF ball_area is Far
THEN set velocity to Fast

Rule 2:
IF ball_area is Medium
THEN set velocity to Medium

Rule 3:
IF ball_area is Close
THEN set velocity to Slow

Rule 4:
IF ball_area is Very Close
THEN set velocity to Stop

Fig. 5 Rules created for the implementation of a fuzzy engine for the attacker's ball tracking.

We hope to make our ball tracking even more accurate and reliable by using the full potential of the fuzzy engine. We want to add more rules and another variable for the dog's current speed.

4. Communication

Another focus of interest for us is using wireless communication between AIBOs. We are working with the Tekkotsu wireless class to create our own code for multiple outgoing connections. Currently we have working code that allows the two attacking AIBOs to maintain bidirectional communication while running our soccer behavior simultaneously. This enables the AIBOs to share information, e.g. robot and ball locations, and allow for multi-agent collaboration to complete more complex strategies. The two attackers inform each other how close they are to the ball and if they are currently trying to grab or kick the ball. In any of these cases the dog further away from the ball will be thrown into a wait state. This state makes the dog get about four feet away from the ball, and the dog will readjust itself if it is too close or too far when it is thrown into this state.

With this communication the dogs can work together and most importantly not work against each other by fighting over the ball. We hope to further utilize this communication by implementing localization within the attackers so that they can inform each other where the ball currently is.

Currently we are using TCP to send character arrays over the network. Each dog sets up a sending network and a receiving network over specified ports. This will be converted to UDP to comply with the rules of RoboCup 2007.

5. Localization

One of the major areas which we are researching and developing is localization. We currently have implemented a localization engine into our defender soccer code. The localization engine can calculate the location and orientation of the AIBO on the field, using the distances and directions to a goal and a beacon. The location can be determined

by the two distances alone. Originally both the beacon distance and goal distance was being estimated to determine the angles to the landmarks and the dog's location. After much testing a new method has been developed. Now we are capturing the dog's head pan angle and offsetting it by the horizontal center of the vision object. This has proven to be much more accurate in calculating than estimating two distances. We still have to estimate one distance to complete the triangle, so we are continuing to estimate our distance to the goal. This estimate has proven to be much more reliable than to the beacon distance estimate.

Currently Tekkotsu does not support multiple vision events in a frame. This at first caused a problem for us to successfully identify a beacon. Currently we allow for vision events that may be a beacon to be compared to the last five frames. This allows us to very accurately decipher the left and right beacon, and recognize beacons in general.

Our method of calculating our final location can be seen in Figure 6. Everything about triangles T_1 and T_2 are the same as when the AIBO is on the other side of the goal. The final n_4 answer is n_1-n_2, or the negation of the other case's n_4 answer of n_3-n_1. If the dog is facing the opposite direction it yields an x value that is in the opposite direction of the goal from that of the other case. To make the results for the two cases compatible we need to reverse the sign of the one case's x result. As x is derived from $\sin(n_4)$, which will be negated if n_4 is negated, and y is derived from $\cos(n_4)$, which will be unaffected if n_4 is negated, we can solve this by using the exact same calculations for both cases.

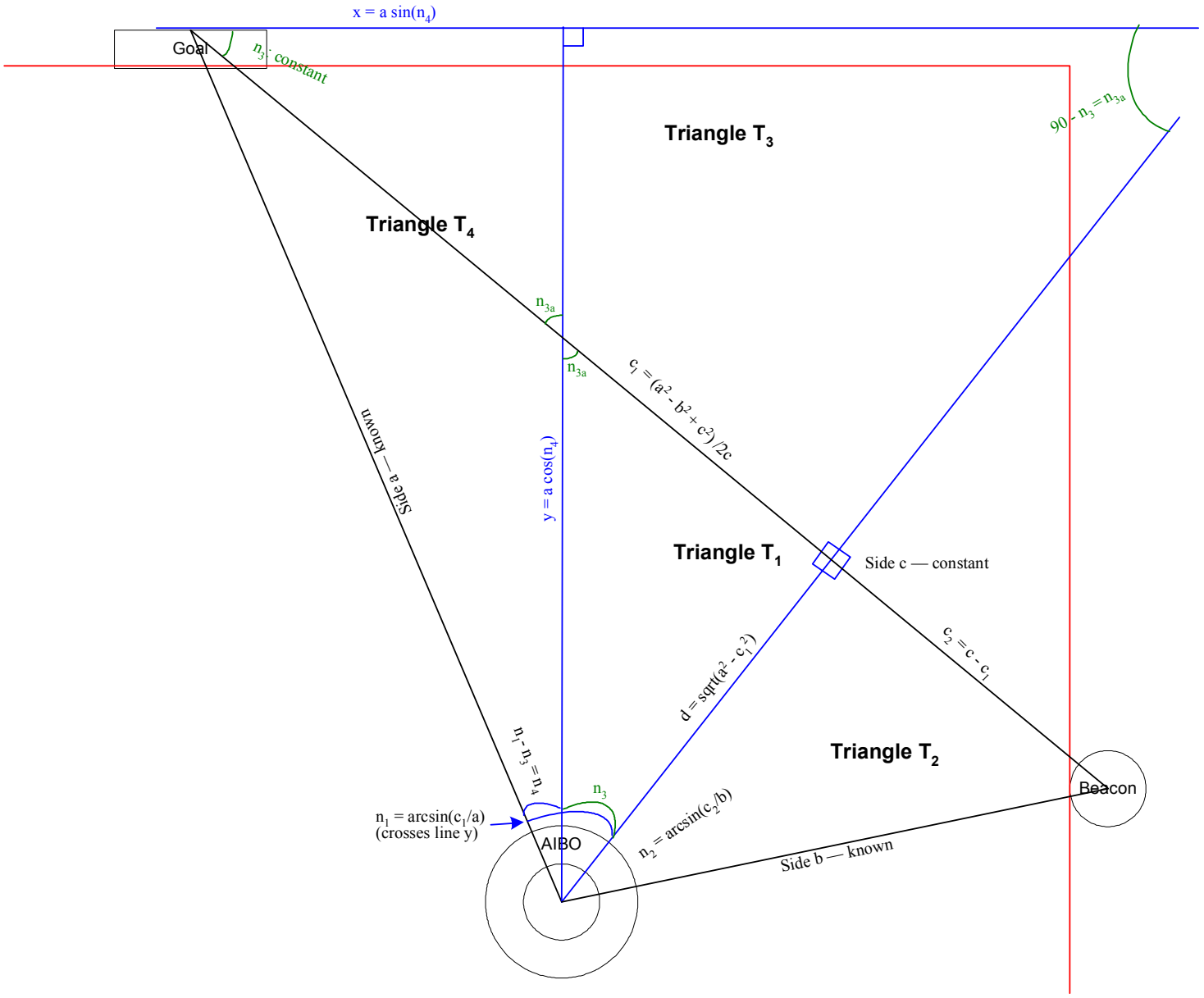

Fig. 6 Representation of the mathematical properties used to calculate localization.

6. Development Tools

In addition to improving and extending Tekkotsu on-board the AIBO we have developed a series of remote processing tools to aid AIBO development. This system, PyTekkotsu, runs alongside Tekkotsu and helps developers prototype and debug behaviors, capture and manipulate motions, run simulations, track the state of the dog, etc. It is written in the Python language so an interactive interpreter can be used to inspect the status or run arbitrary code on the AIBO. PyTekkotsu also includes many utilities to simplify common tasks such as updating objects, changing configurations, and swapping components. PyTekkotsu has been openly released with documentation.

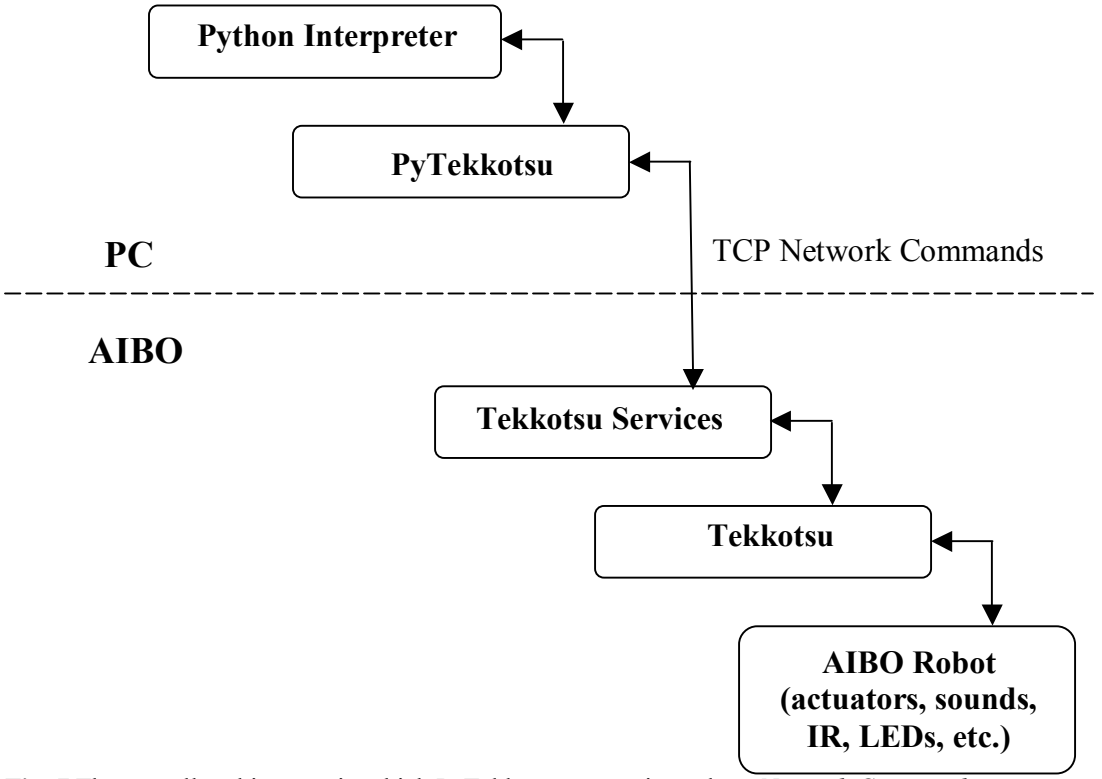

Fig. 7 The overall architecture in which PyTekkotsu comes into play. *Network Commands* are sent between *PyTekkotsu and the Tekkotsu Services.*

PyTekkotsu works exactly the same way as the Java ControllerGUI used by Tekkotsu. It sends commands over the network. Commands are sent over the network to the available Tekkotsu services that are running on the AIBO, e.g. main controller, walk controller, and head controller. PyTekkotsu does not have access to any of the Tekkotsu internals. PyTekkotsu only talks to the simplest of the Tekkotsu services. It does not have access to Tekkotsu's EventRouter, so it cannot listen for button presses or vision events. This is one of the biggest limitations of PyTekkotsu.

7. Conclusion

In conclusion we have made much progress over the last year. We have tested and implemented a localization engine as well as completed work on wireless communication between the attackers. Also, the attackers can now use each other's information to avoid collisions. A fuzzy logic engine has been created, implemented, and calibrated to greatly improve the accuracy and reliability of our ball tracking. In the next year we expect to have all four dogs using localization and wireless communication. We also have many future plans to improve our behaviors and introduce more complex processes.

Acknowledgments. The authors would like to acknowledge Dr. C.J. Chung who has been the LTU Blue Devils Project Advisor since summer 2005. Dr. Chung and Nathaniel Johnson have taken the time to revise and improve this paper. Dr. Bindschadler has supported the LTU AIBO team since its creation as the Mathematics and Computer

Science Department Chair. He is also the LTU AIBO team's senior project advisor for spring 2007.

References

1. Sony, "Sony AIBO." Sony. 25 Feb 2007 <http://www.sony.net/Products/aibo/index.html>.
2. "Tekkostu: Homepage." Tekkotsu. 23 August 2006. Carnegie Mellon University. 23 Feb 2007 <http://www.cs.cmu.edu/~tekkotsu/>.