Software for Automation of Real-Time Agents, Version 2

Version 2 of Closed Loop Execution and Recovery (CLEaR) has been developed. The previous version was reported in “Software for Automation of Real-Time Agents” (NPO-21040), NASA Tech Briefs, Vol. 26, No. 7 (July 2002), page 34. To recapitulate: CLEaR is an artificial intelligence computer program for use in planning and execution of actions of autonomous agents, including, for example, Deep Space Network (DSN) antenna ground stations, robotic exploratory ground vehicles (rovers), robotic aircraft (UAVs), and robotic spacecraft. CLEaR automates the generation and execution of command sequences, monitoring the sequence execution, and modifying the command sequence in response to execution deviations and failures as well as new goals for the agent to achieve. The development of CLEaR has focused on the unification of planning and execution to increase the ability of the autonomous agent to perform under tight resource and time constraints coupled with uncertainty in how much of resources and time will be required to perform a task. This unification is realized by extending the traditional three-tier robotic control architecture by increasing the interaction between the software components that perform deliberation and reactive functions. The increase in interaction reduces the need to replan, enables earlier detection of the need to replan, and enables re-planning to occur before an agent enters a state of failure.

This program was written by Tara Estlin, Daniel Gaines, and Gregg Rabideau of Caltech for NASA’s Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

Software for Optimizing Plans Involving Interdependent Goals

A computer program enables construction and optimization of plans for activities that are directed toward achievement of goals that are interdependent. Goal interdependence is defined as the achievement of one or more goals affecting the desirability or priority of achieving one or more other goals. This program is overlaid on the Automated Scheduling and Planning Environment (ASPEN) software system, aspects of which have been described in a number of prior NASA Tech Briefs articles. Unlike other known or related planning programs, this program considers interdependences among goals that can change between problems and provides a language for easily specifying such dependences. Specifications of the interdependences can be formulated dynamically and provided to the associated planning software as part of the goal input. Then an optimization algorithm provided by this program enables the planning software to reason about the interdependences and incorporate them into an overall objective function that it uses to rate the quality of a plan under construction and to direct its optimization search. In tests on a series of problems of planning geological experiments by a team of instrumented robotic vehicles (rovers) on new terrain, this program was found to enhance plan quality.

This program was written by Marco Quadrelli of Caltech for NASA’s Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

Custom Sky-Image Mosaics From NASA’s Information Power Grid

YourSkyG is the second generation of the software described in “yourSky: Custom Sky-Image Mosaics via the Internet” (NPO-30556), NASA Tech Briefs, Vol. 27, No. 6 (June 2003), page 45. Like its predecessor, yourSkyG supplies custom astronomical image mosaics of sky regions specified by requesters using client computers connected to the Internet. Whereas yourSky constructs mosaics on a local multiprocessor system, yourSkyG performs the computations on NASA’s Information Power Grid (IPG), which is capable of performing much larger mosaicking tasks. (The IPG is high-performance computation and data grid that integrates geographically distributed...
A computer program implements two extensions of ANTLR (Another Tool for Language Recognition), which is a set of software tools for translating source codes between different computing languages. ANTLR supports predicated-LL(k) lexer and parser grammars, a notation for annotating parser grammars to direct tree construction, and predic- cated tree grammars. ["LL(k)" signifies "left-right, leftmost derivation with k tokens of look-ahead," referring to certain characteristics of a grammar.] One of the extensions is a syntax for tree transformations. The other extension is the generation of tree grammars from annotated parser or input tree grammars. These extensions can simplify the process of generating source-to-source language translators and they make possible an approach, called “polyphase parsing,” to translation between computing languages. The typical approach to translator development is to identify high-level semantic constructs such as “expressions,” “declarations,” and “de- finitions” as fundamental building blocks in the grammar specification used for language recognition. The polyphase approach is to lump ambiguous syntactic constructs during parsing and then dis- ambiguate the alternatives in subsequent tree transformation passes. Polyphase parsing is believed to be useful for generating efficient recognizers for C++ and other languages that, like C++, have significant ambiguities.

This program was written by Loring Craymer of Caltech for NASA’s Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (818) 393-2827. Refer to NPO-30565.

**ANTLR Tree Grammar Generator and Extensions**

A computer program implements two extensions of ANTLR (Another Tool for Language Recognition), which is a set of software tools for translating source codes between different computing languages. ANTLR supports predicated-LL(k) lexer and parser grammars, a notation for annotating parser grammars to direct tree construction, and predicated tree grammars. ["LL(k)" signifies "left-right, leftmost derivation with k tokens of look-ahead," referring to certain characteristics of a grammar.] One of the extensions is a syntax for tree transformations. The other extension is the generation of tree grammars from annotated parser or input tree grammars. These extensions can simplify the process of generating source-to-source language translators and they make possible an approach, called “polyphase parsing,” to translation between computing languages. The typical approach to translator development is to identify high-level semantic constructs such as “expressions,” “declarations,” and “definitions” as fundamental building blocks in the grammar specification used for language recognition. The polyphase approach is to lump ambiguous syntactic constructs during parsing and then disambiguate the alternatives in subsequent tree transformation passes. Polyphase parsing is believed to be useful for generating efficient recognizers for C++ and other languages that, like C++, have significant ambiguities.

This program was written by Loring Craymer of Caltech for NASA’s Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (818) 393-2827. Refer to NPO-30565.