CASL

Cognitive Autonomous Systems Laboratory **Leonardo**
Department of Computer and Information Science
Linköping University, Linköping, Sweden

Erik Sandewall

Facilities in Leonardo

This project memo pertains to the development of the Leonardo system.
Identified as PM-leonardo-011, it is disseminated through the CAISOR website
and has URL `http://www.ida.liu.se/ext/caisor/pm-archive/leonardo/011/`

Related information can be obtained via the following www sites:

CAISOR website:	`http://www.ida.liu.se/ext/caisor/`
CASL website:	`http://www.ida.liu.se/ext/casl/`
Leonardo system infosite:	`http://www.ida.liu.se/ext/leonardo/`
The author:	`http://www.ida.liu.se/~erisa/`
Date of manuscript:	2010-05-21

Chapter 1

Administration of Current Agent Contents

The basic and general-purpose command verbs in the Session Command Language are those that operate on the textual and persistent manifestation of the knowledgebase, that is, on entityfiles and knowledgeblocks, and on the individual entities that are contained there. The following is the list of such commands.

This list is incomplete and will be extended from time to time. Recall that although the full representation of a command is a record expression, i.e. it is surrounded by square brackets, but these square brackets may be omitted when the command is entered to the command loop.

1.1 Knowledgeblock Level Commands

Each knowledgeblock is represented as an entity whose name must end with `-kb`. By default, those entityfiles that make a knowledgeblock called `myblock-kb` shall be located in a directory called `Myblock` that is an immediate subdirectory of the agent directory. The entity whose name ends with `-kb` represents both an entityfile with the same name i.e. `myblock-kb` and the entire knowledgeblock. The kb entityfile contains entities that provide the access path to each of the entityfiles in the knowledgeblock.

The following are the most important operations on knowledgeblocks.

`[crek` ***kb***`]`

Create a knowledgebase called ***kb*** with initial, minimal contents. For example, `crek runners-kb` will create a directory called `Runners` containing one single entityfile, called `runners-kb,` and it will register the new knowledgebase in `kb-catal` which is the register of all knowledgebases in the agent at hand. Ω

`[defk` ***kb***`]`

Similar to `crek`, but that operation initializes the new knowledgeblock to having empty contents, which the present operation does not. It only introduces ***kb*** in `kb-catal` without affecting its contents. This is for use if a full

directory for ***kb*** has been or will be copied in from outside as a directory level operation. Ω

`[setk` ***kb***`]`

Assign ***kb*** as the *current knowledgeblock.* This setting is used by the `crefil` command for determining the choice of knowledgeblock where a new entityfile is to be located. Ω

`[loadk` ***kb***`]`

Loads the knowledgeblock ***kb*** using the following procedure.

- Load the entityfile with the name ***kb*** using its `.leos` version.
- Load recursively the knowledgeblocks listed under the `requires` attribute of ***kb.***
- Load the entityfiles listed under the `mustload` attribute of ***kb.***
- Execute the function in the `end-startup-proc` property of ***kb.*** This function is usually a no-op but in some cases it has a nontrivial definition.

Ω

Each knowledgeblock file contains, as its second item, the definitions for three Lisp functions that are attached to the knowledgeblock entity, and which provide handles for operations to be performed at the beginning of the loading of the knowledgebase, at the end of its loading, and (not yet implemented) at the closing of a session where this knowledgeblock has been loaded.

The `preferred-directory` attribute of a knowledgeblock provides a handle for redefining the location of additional entityfiles in the block. The default value for `myblock-kb` is `"Myblock/"` i.e. it specifies the location relative to the top-level directory of the agent in question. Redefining it will affect the location of additional files that are created using the `crefil` command, but it will not affect the location of existing member files.

1.2 Entityfile Level Commands

The structure of entityfiles has been described in the memo KRF Overview, Chapter 5. The following are the major commands for entityfiles.

`[crefil` ***ef***`]`

Creates a new entityfile called ***ef*** and locates it in the current knowledgeblock as set by the `setk` command. Ω

`[curfil` ***ef***`]`

Sets ***ef*** as the current entityfile, as used by the commands `loadf, writef, sortf` and others. Ω

`[loadfil` ***ef***`]`

Loads the entityfile ***ef*** from its `.leo` manifestation and into the current session. Ω

`[loadf ]`

Loads the current entityfile as set by the `curfil` command, and in the same way as for the `loadfil` command. Ω

`[writefil` ***ef***`]`

Write the physical file for the entityfile ***ef*** using the information in the current session. Ω

`[writef ]`

Write the physical file for the current entityfile using the `writefil` operation. Ω

`[updfil` ***ef***`]`

Updates the entityfile ***ef*** i.e. loads it and then writes it again. It is always loaded from the `.leo` version, even if it is a `datafile.` Care must be exercised since the loading operation catches errors but if an error occurs then the writing operation proceeds anyway, which may destroy the contents of the entityfile. Ω

`[updf ]`

Updates the current entityfile using the `updfil` operation. Ω

`[sortf ]`

Sorts the list of contents of the current entityfile in alphabetical order. If the file is partitioned into sections then each section is sorted separately and the order of the sections is preserved. The command does not load or write the entityfile. It is not entirely robust, so it is recommended to take a backup of the argument file before doing the sorting. Ω

1.3 Entity Editing Commands

`[creobj` ***obj typ***`]`

Initiates an entity called ***obj*** of type ***typ*** and makes it the current entity. If ***typ*** has a value for the property `create-proc` then this value shall be a function that is applied to ***obj*** as its single argument. Ω

`[curobj` ***obj***`]`

Sets ***obj*** as the current entity for the purpose of the `edo` operation. Ω

`[edobj` ***obj***`]`

Allows the user to edit the contents of the entity ***obj*** by the following steps: it writes the attributes and properties of ***obj*** to a temporary file, invokes the user's preferred text editor on this file, allows the user to edit it, and then reads the file back into the session. Ω

`[edo ]`

Allows the user to edit the contents of the current entity using the `edobj` operation. Ω

Chapter 2

Administration of Leonardo Individuals

2.1 Reproduction

Available commands:

`[breed` ***name***`]`

Creates a new agent with the given name in the same individual as the current agent, that is, located sideways from it. The current session is damaged and should be terminated after the breed operation has been performed. The new agent is made as an offspring of the present one, i.e. its identifier is obtained from the identifier of the present agent by appending `-nnn` where `nnn` is a threeplace number reflecting the number of this offspring relative to other, previous offspring from the present agent. Ω

2.2 Cloning

Available commands:

`[makclone ]`

This operation will convert the present agent to a boilerplate for cloning. It should be a remus agent. After this operation, it will start by establishing itself by prompting the user and by looking around in the host where it is executing. Please recall that the following additional measures must be taken to prepare a clone:

- Change the name of startleo.bat to startleo.batt (in order to avoid irritating virus filters)
- Combine it with a minimal Indivmap directory
- (Optional) zip it

  If you wish to proceed with clone conversion, type yes and Return Ω